PARTITIONED PARTICLE FILTERING FOR TARGET TRACKING IN VIDEO SEQUENCES

By
Markus Smuts Louw

SUBMITTED IN FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
AT
UNIVERSITY OF CAPE TOWN
CAPE TOWN, SOUTH AFRICA
FEBRUARY 2004

UNIVERSITY OF CAPE TOWN
DEPARTMENT OF
ELECTRICAL ENGINEERING

The undersigned hereby certify that they have read and recommend to the Faculty of Engineering and the Built Environment for acceptance a thesis entitled "**Partitioned particle filtering for target tracking in video sequences**" by **Markus Smuts Louw** in fulfillment of the requirements for the degree of **Master of Science**.

Dated: February 2004

Supervisor: Prof G de Jager
Dr Frederick Nicolls

Readers:

UNIVERSITY OF CAPE TOWN

Date: **February 2004**

Author: **Markus Smuts Louw**

Title: **Partitioned particle filtering for target tracking in video sequences**

Department: **Electrical Engineering**

Degree: **M.Sc.** Submitted: **February** Year: **2004**

Signature of Author

Table of Contents

Declaration

I declare that this thesis is my own unaided work. It is being submitted for the degree of Master of Science in Engineering at the University of Cape Town. It has not been submitted before for any degree or examination at this or any other academic institution.

Markus Smuts Louw

Signature of Candidate

Cape Town

February, 2004

Acknowledgements

The author would like to give his thanks to the solid support of the members in the Digital Image Processing Group at UCT. This work spans both years and continents, and is the richer for it. Thanks to Jerome for his valuable input with regards to colour analysis, and to Bruno, Mathew and Prof. de Jager, and also in particular to Dr. Fred Nicolls, who provided numerous suggestions and good perspective on the problem. The author also gives thanks for the financial support given by the National Research Foundation (NRF), and by DeBeers Technology Group (DebTech).

Also thanks go to Dene and to Julie, who are my wonderful family.

Abstract

A partitioned particle filtering algorithm is developed to track moving targets exhibiting complex interaction in a static environment, in a video sequence. The filter is augmented with an additional scan phase, which is a deterministic sequence which has been formulated in terms of the recursive Bayesian paradigm, and yields superior results. One partition is allocated to each target object, and a joint hypothesis is made for simultaneous location of all targets in world coordinates. The observation likelihood is calculated on a per-pixel basis, using sixteen-centered Gaussian Mixture Models trained on the available colour information for each target. Assumptions about the behaviour of each pixel allow for the improvement under certain circumstances of the basic pixel classification by smoothing, using Hidden Markov Models, again on a per-pixel basis. The tracking algorithm produces very good results, both on a complex sequence using highly identifiable targets, as well as on a simpler sequence with natural targets. In each of the scenes, all of the targets were correctly tracked for a very high percentage of the frames in which they were present, and each target loss was followed by a successful reacquisition. Two hundred basic particles were used per partition, with an additional one hundred augmented particles per partition, for the scan phase. The algorithm does not run in real-time, although with optimization this is a possibility.

Chapter 1

Introduction

The problem of target tracking in video sequences is a very advanced and complex area of signal processing theory. Although the former seems far removed from the usual scope of the latter, we essentially seek to analyze a two dimensional signal through time and to identify within that signal, sub-signals, or "targets" which may themselves vary in their characteristics, and may even interact with one another and with the embedding signal. The embedding signal would then constitute the image of the background scene within which the objects are moving. Due to the complexities of the interactions, the most effective methods which have been developed maintain joint hypotheses about the location as well as the characteristics of the targets to be found, with respect to the background scene, thus containing and explaining the observed target occlusions.

The applications of target tracking in general which are of importance in engineering in today's world are many and varied, ranging from military imaging with weapon target detection and tracking, to radar target tracking, to security and surveillance applications. Target tracking in video sequences may be used for an even wider range, since the information content in a video sequence is so much higher than that of radar or sonar or most other sensor devices, passive or active. The main areas of application of target tracking in video sequences are:

- Automatic computer controlled surveillance
- Behaviour analysis
- Video conferencing and speaker detection and identification
- Gait analysis for medical applications
- Video compression with region of interest identification.

There are even some unusual applications for target tracking such as analysis of social insect (and other animal) behaviour, and the analysis of player style and team strategy in sports.

In this thesis we implement and explore the performance of partitioned particle filtering on the problem of tracking with a high number of identifiable target objects. In this case the targets are people, moving and interacting in a static scene. What makes this problem specification more difficult than usual is the high amount of occlusion which results from the interaction of the high number of people. Occlusion and noise are generally the biggest problems in any target tracking implementation, and the tracker's robustness is a measure of how well it recovers from these phenomena, when it loses the target.

Particle filtering has only as recently as 1993 surfaced in the domain of computer vision, where before this, the computational expense of this form of filtering was prohibitive for online or even offline tracking applications. The advantage which the particle filter has over other types of filters (Kalman, Extended Kalman, etc.) is that it allows for a state space representation of any distribution. It also allows for nonlinear, non-Gaussian dynamical and observation models, and nonlinear, non-Gaussian process and observation noises. This results in superior performance in areas where the dynamics and observations are in fact nonlinear and non-Gaussian and where the process and observation noises are essentially highly non-Gaussian, as with image clutter. The partitioned particle filter was first proposed in 1999, and allows for the partitioning of the state space for a more intelligent sampling strategy. The computational efficiency gained through this method is of increasingly greater benefit as the dimensionality of the state space increases, hence its applicability to the problem of tracking a large number of targets via a joint hypothesis state vector, which exists in a space of high dimension.

1.1 Intelligent environments

The creation of a so called intelligent environment, in which the locations of objects of interest and of people is known at every moment is a goal in research at academic institutions and in the industry. The ways of discovering these locations may be active, through the use of transmitters, or passive, through the use of sensor devices.

Computer vision provides an excellent solution to the problem of discovering and tracking the locations of targets in a scene, with various algorithmic approaches described in the next chapter. There are many reasons why one may wish to create an automated monitoring process, which can report at any time where all the targets in the database are in the intelligent environment. Once the locations of targets, which in this application are people,

are known, further analysis may be performed on this data, for example behavioural analysis algorithms may be run on the location data, and on the information which subsequent sub-location and orientation algorithms reveal. Suspicious or criminal behaviour may be detected the moment it occurs, if a target is not where it should be or if it behaves in an unusual way.

1.2 Problem statement

In certain intelligent environments, particularly ones involving high security, it may be possible and highly useful to restrict the people in the scene to wearing particular types of clothing, to assist in the performance of the tracking algorithm. The goal is the monitoring of the interaction of the people in the environment, with the environment, and with one another. Depending on the quality of the tracking required, targets may also wear natural clothing, which would make a colour model tracking method such as this one operate less effectively, but within reason, i.e. within the parameters of usable tracking performance. Therefore we will also investigate (although in less depth) the performance of this tracker on naturally-coloured targets. For the target recognition stage, it is possible via blue screen technology to extract automatically segmented colour information of the target, prior to entry into the intelligent environment, and in this implementation, this is the stage which is simulated via the manual segmentation of the targets. We also assume that the cameras used to monitor these scenes, and which are the input for the tracker, are static. This allows us to discriminate more easily between background and foreground regions.

The emphasis in this thesis is to solve the problem of developing the best tracking algorithm for such a controlled environment, which may later be optimized for speed and real-time performance. A real-time tracker based on this algorithm may then be connected to a network of similar trackers, which would pass initialization information to each of the trackers receiving a new target in its range of detection.

1.2.1 Datasets and manual labelling

To establish an accurate empirical metric against which the performance of the tracker here developed may be compared with alternative solutions, each of the pixels in each of the frames in each of three sequences has been labelled manually. This is done to benchmark the algorithm, and although it is an unusual investment of time, (in the tracking literature, the performance of tracking algorithms is usually described qualitatively), the author feels it was necessary for true evaluation of tracking performance. Due to the time consuming nature of this manual labelling, only three sequences were so labelled.

Figure 1.1: A frame from the indoor benchmark sequence.

1.2.2 Indoor (benchmark) sequence

To test the algorithm which is here developed, we use three test sequences. The first test sequence is an indoor sequence, of seven people wearing highly coloured clothing, which makes each of the targets highly identifiable, provided that they are not occluded. The people arrive from four access points, and it is at these access points that the tracker for the intelligent environment may be instructed to scan at regular intervals for the arrival of new targets. The people arrive in a random order, and walk in independent random circles around the room, deviating from these circles regularly to jump, crouch, or otherwise. The interaction and occlusion between these people is designed to be very complex, far more than for natural scenes, and as complex a situation as a tracker might be expected to track successfully. The sequence is 460 frames long, and the people leave in a different order from that in which they arrived. This indoor sequence is the benchmark sequence in the sense that it reveals the true conditions under which the algorithm is intended to operate, and therefore more analysis has been applied to the performance of the tracker in this scenario. In Fig. 1.1 we can see a typical frame from the indoor sequence.

1.2.3 Outdoor sequences

The algorithm is also tested on two outdoor sequences. In these sequences, there are four people which are naturally coloured (wearing normal clothes), and there is a corresponding increased difficulty in distinguishing these targets from one another. The motion of the

Figure 1.2: A frame from the first outdoor sequence

targets is also simple and natural, and the tracker's performance in these sequences is tested to evaluate its performance in this scenario. These outdoor sequences are included as an extended investigation into the performance of the tracker in other scenarios, but the performance has not been analyzed in the same depth as the indoor sequence. In Fig. 1.2 and Fig. 1.3 we can see typical frames from each of the two outdoor sequences.

1.3 Breakdown of thesis

In **chapter 2**, the current and past literature on the subjects of target tracking, colour models, Hidden Markov Models and particle filtering is reviewed, as well as some of the variants on the basic particle filtering scheme which are not strictly Bayesian in their formulation, but which still provide good, often excellent tracking performance.

In **chapter 3**, the formulation of the basic particle filter is given, and the notation is given which shall be used in this thesis with regards to particle filtering.

In **chapter 4**, the method of observation of the state is described, as well as the reasons for the choice of colour space within which this observation process is performed. The methods of training and using Gaussian Mixture Models for the classification of pixels in the images of the video sequence is described.

In **chapter 5**, the effects of using Hidden Markov Models on a per-pixel basis to improve the classification of the pixels given by the Gaussian Mixture Models of Chapter 3 are explored.

Figure 1.3: A frame from the second outdoor sequence.

In **chapter 6**, importance sampling, partitioned sampling, and the scan phase, which is a novel implementation in this thesis, are described.

In **chapter 7**, the results of the combination of the sampling and observational techniques in this implementation are described and compared against manually labelled data. The basic particle filter, the partitioned particle filter, and the partitioned particle filter with the additional scan phase, are all compared.

In **chapter 8**, we conclude with a summary of the results and of some of the interesting effects that were observed. Also given are directions for possible future areas of research, extending this work.

Although many of the latest particle filtering algorithms developed in the last few years run in real-time, the software developed for this application does not. The framework within which this application was developed, was optimized for extensibility and not for speed. It is only in the more recent years that the computational power required to use particle filtering in target tracking in video sequences has been possible at all, and it is usually difficult to make video based target tracking implementations which can run in real-time. The recent developments in particle filtering for tracking in video sequences often reduce the computational expense at the observation stage by using an analysis of the joint hypothesis to select only a part of the image region within which to perform the observation, although this is not always valid as one should use all the available observation data at each time step if one is to develop the best hypothesis. One particularly effective method for doing this is discussed in Chapter 4, and involves computing in advance the

probability for the classification of each pixel as a background pixel, and dividing through by this value when calculating a foreground pixel's probability, resulting in all of the image data being implicitly taken into consideration [1]. In general and in this implementation we may say that iterating through each pixel for each frame once per particle is certainly the most expensive stage of the filter, and in this implementation, where there is one state space partition created per target object, the computation required is multiplied by the number of partitions. With state of the art equipment, and highly optimized code, it is likely that this algorithm could run in real-time, however that is not the goal of this research.

Chapter 2

Literature Review

In this chapter we review some important literature in the various methods of target tracking, using linear and nonlinear filters. Methods for target location are reviewed, as well as methods for target hypothesis evaluation.

2.1 Kalman Filter based trackers

In [21], published in 1970, Bar-Shalom and Fortmann develop the Probabilistic Data Association Filter (PDAF), in which a number of extensions to the Kalman filter are described for dealing with a multi-modal Gaussian observation density, and a linear Gaussian process density. Later in [23], their Joint PDAF (JPDAF) is developed, and the state density is then also represented by a multi-modal Gaussian distribution. This is relevant in the general case where there is uncertainty in the target labelling, i.e. after the target detection stage, when the targets need to be identified. In this implementation the targets are highly colourized and identifiable, but such data association methods can be incorporated into a particle filter framework when the targets are capable of being confused with one another.

In [16], published in 1995, Goncalves et al. track the pose of a human arm in a monocular image seqence, where the shoulder remains fixed, using a Kalman filter based approach. Rehg and Kanade [17] also use a Kalman filter approach for self occluding articulated objects, to track a hand.

Intille et al. in [18], published in 1997, develop a real-time tracking algorithm which can track multiple objects in a known, closed environment, using contextual information which allows for the adaptive weighting of the features used to make the matching correspondences. The features used are blob size, colour, velocity and position. Simple Euclidean distances are used for the position differences and the colour-average distances between blobs and objects. This implementation has problems when dealing with blob merging, inaccurate

scenario where many points may belong to each target, all among background clutter.

In [41], Torma and Szepesvari develop LS-N-IPS (Local Search N-Interacting Particle System), replacing the dynamics stage of the standard particle filter with a local search step. Each particle is thus refined analytically based on the information yielded in the observation stage, which is a B-spline perpendicular distance measure similar to that used by Isard in [7], and MacCormick and Blake in [32]. A uniform convergence property for LS-N-IPS is also proven.

In [42], published in 2003, Khan et al. track multiple targets (social insects) using a Markov Random Field as the prior for a motion model. A single joint particle filter for all targets is discarded in favour of a particle filter for each target, which is reported to yield superior results. Each filter is prevented from being in a similar state to any other via a Markov Random Field, which is a factor in the observation for each filter.

Pitt and Shepherd in [44] (published in 1997) developed the auxiliary particle filter, and tested it on simulated range data, where it outperformed the standard particle filter. Two weighted bootstraps are performed per time step. In the first, particles are weighted according to some function of the dynamics distribution rather than the sample drawn from the dynamics. This intermediate distribution is then resampled and weighted according to the true observation for the particle.

In [46], published in 2003, Bruno develops a mixed-state particle filter framework, according to which changes in the object's appearance may be modelled via a state variable and a corresponding method for observing the target depending on its state. The framework is developed for both the Auxiliary Particle Filter as well as the SIR particle filter. The algorithms are tested on synthetic infrared airborne radar data, where both outperform the standard condensation tracker.

Nummiaro et al. in [47] track multiple objects using an adaptive colour based particle filter. The Bhattacharyya coefficient is used to compare the distributions of the observation and the target model. The model is only updated if the current estimated position of the object has an associated probability which is higher than some threshold. This will prevent the target histogram from being updated if the target is temporarily or permanently lost.

In [48], published in 2003, Li and Chue use an adaptive colour histogram technique for tracking human targets. Data is classified as labelled or unlabelled at each time step, and the unlabelled data is used to adjust the colour histogram which is calculated from the labelled data. Unlabelled data contributes in proportion with the probability which is associated with the particle which generated it, and unlabelled data is so classified if its probability measure is below a certain threshold.

Odobez et al. [50] in 2003 remove the independence assumption of the data per frame in

the SIR particle filter, adapting the filter so that dependence between data at successive time steps is taken into account, as is dependence between data and the state at the previous time step. It turns out that only the measurement equation changes. This measurement equation is factorized into the product of a correlation measurement and a shape measurement. In the correlation measurement, the correlation between the data at the target window at the current and previous time steps is taken into account; in the shape measurement, contour distances may be calculated. In this way the need for pre-trained appearance models is bypassed, provided that the tracker is initialized at the correct position. Pre-trained target reference models such as colour histograms may also be used to augment this system, yielding improved results.

In [52], Zhou et al. stabilize the standard tracker using adaptive observation models, a velocity model with adaptive noise variance, and adaptive numbers of particles. The adaptive velocity is computed using a first order linear predictor depending on the previous particle configuration. Their algorithm is tested on vehicles and human faces, and is superior to a standard particle filter.

In [53], published in 2002, Vermaak et al. use a stochastic Expectation-Maximization algorithm to adapt the observation model of the target object (where the observation consists of histogram comparison). In addition, the observation model is adjusted only when the target is both present and in motion. The dynamical model for the translation and scaling of the target follows a Langevin dynamical model, and conjugate Dirichlet priors are used for the observation model to allow for a closed form solution in the Maximization step of the observation adjustment. In a similar implementation to [1], the image data is preprocessed with three isotropic Gaussian filters per pixel, two to reveal the colour, and one to reveal the apparent target motion occurring at that pixel. At each time step the image is scanned for likely target locations, and these locations are used as the proposal distribution in the particle filter.

2.4 Variants on the Particle Filtering algorithm

In [49], published in 2002, Bruno extends the bootstrap filter to include importance sampling, and also uses a 2D non-causal Gauss-Markov random field model to describe the clutter spatial correlation. The tracking is compared to tracking on the same data using the Kalman-Bucy filter, which the extended bootstrap filter then outperforms.

Tacher and Darrel, in [55], published in 2003, develop a Bayesian pose recovery algorithm to track the upper body of a person. Importance sampling and a kinematic modelling are used, and at each stage the current frame is analysed and the pose extracted. The pose from

the previous frame may be used as a constraint on the sampling for the current frame, but the framework does not conform to particle filtering, although according the the authors, the implementation is superior to the basic particle filter with diffusion dynamics.

In [27], published in 1997, Higushi applies principles from genetic algorithms, namely the crossover and mutation operators, to the sample sets which represent the posterior distributions of the standard particle filter at each time step. The sample sets are treated as populations and the samples as individuals, the components of which may be combined with one another (crossover) or altered (mutation) to provide samples for the iteration at the next time step. The new population (sample set) in this implementation was generated after the Sequential Monte Carlo (SMC) resampling stage. The drawback of this implementation is that the genetic modification stage affects the convergence of the algorithm to the true posterior distribution. In addition, the genetic modification of the samples in the population were based on their binary representation, as is often the case in genetic algorithms, however this is inappropriate for high precision, multidimensional problems.

In [28], published in 2002, Tito et al. introduce a new form of the Genetic Particle Filter, called the Genetic Sampling Importance Resampling (GSIR) filter. In this implementation, the genetic operators are capable of working directly on the floating point numbers, and also allow for convergence to the true posterior distribution.

Choo and Fleet in [64], published in 2001, develop a Hybrid Monte Carlo (HMC) filtering algorithm to track people. Each person is tracked by a Monte Carlo Markov-Chain, with the idea that the state space may be explored faster than in a traditional particle filter by creating a fewer number of particles, each of which searches the state space using the posterior gradient, while all the chains taken together, still converge towards the correct posterior. In a 28 dimensional tracking problem, it is found that the HMC filter is several thousand times faster than a traditional particle filter.

2.5 Multi Camera Tracking

In [61], published in 1996, Gavrila and Davis use a generate-and-test strategy to search through pose space for the articulated body tracking, with a robust variant of *chamfer matching* [62] used as a match metric. Areas of interest in the multi-view data are obtained through background subtraction, and the basic pose (major axis of orientation) of the person is extracted from a Principal Component Analysis (PCA) of data points sampled from the region of interest. A constant acceleration model is used with a best fit search technique, and the search space is decomposed in a precursor to partitioned sampling for condensation trackers.

Haritaoglu et al, in [56], published in 1998, use intensity and disparity images obtained from a stereo camera rig. Background pixels are modelled by establishing for each a maximum, minimum and maximum change per frame (intensity only is used, not colour). Objects are then matched using templates, shape analysis and dynamics.

In [57], published in 1998, Darrel et al. combine colour and face detection and the use of a stereo camera system for tracking people. A distinction is made between long, medium and short term tracking, and different data is used to track in these different phases (i.e. face detection for long term tracking and dynamics for short term tracking).

Krumm et al. in [58], published in 2000, locate the targets using stereo information, then use colour to identify them, by means of histogram intersection. Birchfield in [59] (1998) also used histogram intersection as well as intensity gradients to track heads.

In [60], published in 2002, Black et al. use a multi camera system and a pair of Kalman filters per target per camera. One of the Kalman filters tracks the object in 2D image coordinates, while the other simultaneously tracks the object in 3D world coordinates. Information about the observation uncertainty is passed by covariance propagation from the 2D Kalman filter to the 3D Kalman filter to assist its tracking, and viewpoint integration is done by calculations of the homographies (planar mappings) of the views.

Lee et al. in [63], published in 2002, develop a method for integrating analytical inference into the particle filter tracking, which allows subsets of the state space to be updated at any time. Points belonging to the target object are solved for using the multiple-view geometry, and these points are analyzed to find specific body parts within the region in world coordinates. This information is then used to solve for a part of the state space, and the particle filter is then used to search the rest of the state space.

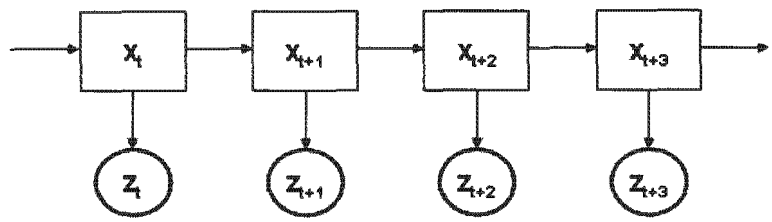

Figure 3.1: Graphical representation of conditional dependencies over four time steps.

This graphical notation for conditional dependence is used extensively in the field of Bayesian networks, and has been used in [50] and elsewhere.

3.1.5 Particle Filter Equations

The rule for the propagation of the state probability distribution in time is:

$$p(\mathbf{X}_t|Z_t) = k_t p(\mathbf{Z}_t|\mathbf{X}_t)p(\mathbf{X}_t|Z_{t-1}),$$

where

$$p(\mathbf{X}_t|Z_{t-1}) = \int_{\mathbf{X}_{t-1}} p(\mathbf{X}_t|\mathbf{X}_{t-1})p(\mathbf{X}_{t-1}|Z_{t-1})d\mathbf{X}_{t-1},$$

in which $p(\mathbf{X}_t|\mathbf{X}_{t-1})$ represents the state dynamics, and may be defined for each time step t, and $p(\mathbf{Z}_t|\mathbf{X}_t)$ represents the observation density at time t. The observation density may also change over time, as the data $\mathbf{Z}_t$ against which the state vector $\mathbf{X}_t$ is to be compared. This latter probability is the answer to the question, *how likely is it that a particular state vector gave rise to the observed data.* The normalizing term k_t may be expanded as

$$k_t = p(\mathbf{Z}_t|Z_{t-1}),$$

and k_t then does not affect our state prediction except via an overall multiplicative factor, which will not affect our state estimation, prediction, or any other operations.

There are a number of ways in which these equations may be carried out algorithmically. Perhaps the most commonly used method is to represent the current state distribution by a weighted set of samples (also known as *particles*): $\{\mathbf{s}_t^{(n)}, \pi_t^{(n)}, c_t^{(n)}\}$, with $n = 1, 2, .., N$ where $\mathbf{s}_t^{(n)}$ represents the n_{th} sample at time t, $\pi_t^{(n)}$ represents the probability weight associated

with that sample, and $c_t^{(n)}$ represents the cumulative probability distribution of the samples up to the sample in question, where we assume some random order has been established on the samples. Such a sample set may then approximate any distribution $P_t(\mathbf{x})$:

$$P_t(\mathbf{X}) = \sum_{i=1}^{N} \pi_t^{(i)} \delta(\mathbf{X} - \mathbf{s}_t^{(i)}),$$

where $\delta(.)$ is the Dirac delta. The question arises how we may sample, or simulate from a distribution represented by such a set of particles, and one way is Factored Sampling. As the number of samples tends to infinity, so does the approximate distribution tend toward the true distribution.

3.1.6 Factored Sampling

Factored sampling represents one way to sample from this distribution. As mentioned earlier each sample is assigned the normalized probability

$$\pi_t^{(i)} = \frac{p_z(\mathbf{s}_t^{(i)})}{\sum_{j=1}^{N} p_z(\mathbf{s}_t^{(j)})},$$

where

$$p_z(\mathbf{s}_t^{(i)}) = p(\mathbf{Z}_t | \mathbf{X}_t = \mathbf{s}_t^{(i)}),$$

which is the observation density given a sample value $\mathbf{X}_t$ and measurement data $\mathbf{Z}_t$.

To perform factored sampling, we choose from this prior, with replacement, sample $\mathbf{s}_{t-1}^{(j)}$ with probability π_{t-1}^{j}. For each sample thus chosen, we sample again from the dynamical distribution $p(\mathbf{X}_t | \mathbf{X}_{t-1} = \mathbf{s}_{t-1}^{(j)})$, to generate a sample $\mathbf{s}_t^{(j)}$, and this sample is then evaluated against the observation density $p(\mathbf{Z}_t | \mathbf{X}_t = \mathbf{s}_t^{(j)})$ to find its unnormalized probability.

3.1.7 Algorithm for particle filtering using factored sampling

The following algorithm will generate from the prior, represented by the sample set $\{\mathbf{s}_{t-1}^{(n)}, \pi_{t-1}^{(n)}\}$, with $n = 1, 2, .., N$ for time $t-1$, a posterior distribution, represented by the sample set $\{\mathbf{s}_t^{(n)}, \pi_t^{(n)}\}$, with $n = 1, 2, .., N$ for time t.

For each time step, construct the n_{th} of N new samples as follows:

1. Select a sample $\mathbf{s}_{t-1}^{\prime(n)} = \mathbf{s}_{t-1}^{(j)}$ with probability π_{t-1}^{j}.
2. Predict by sampling $\mathbf{s}_t^{(n)}$ from the distribution:

$$p(\mathbf{X}_t|\mathbf{X}_{t-1} = \mathbf{s}^{\prime(n)}_{t-1})$$

3. Calculate the observation probability using the observation data for that timestep $\mathbf{Z}_t$, and weight the particle accordingly. So

$$\pi_t^{(n)} = p(\mathbf{Z}_t|\mathbf{X}_t = \mathbf{s}_t^{(n)})$$

After all N samples have been generated with their corresponding probabilities, the sample probabilities across the sample set are normalized so that:

$$\sum_{i=1}^{N} \pi_t^{(i)} = 1$$

We may also calculate the cumulative distribution across the sample set for ease of sampling in the next time step.

Each sample may thus be stored as a tuple $(\mathbf{s}_t^{(n)}, \pi_t^{(n)}, c_t^{(n)})$ after all samples have been evaluated where:

$$c_t^{(1)} = \pi_t^{(1)}$$
$$c_t^{(n)} = c_t^{(n-1)} + \pi_t^{(n)} \quad \{n = 2, 3, .., N\}$$

The benefit is that to generate a sample from a sample set so represented, we may randomly generate a number r from the uniform distribution between zero and one, and then search in increasing order through the sample set until we come across a sample whose cumulative probability weighting $c_t^{(n)}$ is greater than r. This is then a fair (approximated) sample from the distribution which the sample set approximates.

3.1.8 Processing all the samples

Another formulation of the algorithm to represent the particle filter equations is to evaluate all of the samples which make up the prior. Although this is generally inadvisable, since we would prefer to spend our computation on samples which are closer to the peaks of the prior distribution, this formulation is still valid and is also helpful in understanding Importance

Sampling which we discuss later.

The following algorithm will generate from the prior, represented by the sample set $\{\mathbf{s}_{t-1}^{(n)}, \pi_{t-1}^{(n)}\}$, with $n = 1, 2, .., N$ for time $t-1$, a posterior distribution, represented by the sample set $\{\mathbf{s}_t^{(n)}, \pi_t^{(n)}\}$, with $n = 1, 2, .., N$ for time t.

For each time step, construct the n_{th} of N new samples as follows:

1. Select the n_{th} sample from the prior. $\mathbf{s}_{t-1}^{\prime(n)} = \mathbf{s}_{t-1}^{(n)}$
2. Predict by sampling $\mathbf{s}_t^{(n)}$ from the distribution:

$$p(\mathbf{X}_t | \mathbf{X}_{t-1} = \mathbf{s}_{t-1}^{\prime(n)})$$

3. Measure and weight the new sample according to the observation data for that timestep $\mathbf{Z}_t$ and the source sample's probability in the prior. So

$$\pi_t^{(n)} = p(\mathbf{Z}_t | \mathbf{X}_t = \mathbf{s}_t^{(n)}) \pi_{t-1}^{(n)}$$

After all N samples have been generated with their corresponding probabilities, we then normalize the sample probabilities across the sample set so that:

$$\sum_{i=1}^{N} \pi_t^{(i)} = 1$$

Actually, we also require for this algorithm to be effective a resampling step after the third step, but we will discuss this in greater depth in Chapter 6. We can see that this algorithm is similar to the Factored Sampling algorithm, but the first step has become redundant since all samples are processed. Thus we need to maintain each sample's prior probability as a factor in its posterior probability in step 3.

3.1.9 Moments

If we are interested in estimating the moments of the posterior distribution, we may calculate them directly from the sample set as:

$$E[f(\mathbf{X}_t)] = \sum_{n=1}^{N} \pi_t^{(n)} f(\mathbf{s}_t^{(n)}).$$

For example to establish the mean state we could set

$$f(\mathbf{X}_t) = \mathbf{X}_t.$$

More often we are interested in the maximum a posteriori (MAP) value in the posterior distribution, which corresponds to the sample in the posterior sample set which has the highest associated probability, or

$$m = \underset{\mathrm{j}}{\operatorname{argmax}} \; P(\mathbf{Z}_t|\mathbf{X}_t = \mathbf{s}_t^{(j)}),$$

then $\mathbf{s}_t^{(m)}$ is the sample with the highest observation probability and for our purposes the MAP state estimate for a particular time step t. Although it can be argued that when we use the Minimum Mean Square Error (MMSE), which is the first moment of the state, the estimate is more robust, it was judged in this implementation that the MAP estimate gives visually preferable results for the state. To show this, consider a minimal case, where two particles are used to track an object. When the MMSE estimate is used, if a correct particle has a probability only slightly greater than the second particle which represents an incorrect hypothesis (this may occur for example when a particle is tested against a target which is similar to the actual target), the MMSE estimate will be a weighted average of these two particles. But because the weights are similar, the estimate exists in space somewhere between the true target and the false one, and is entirely useless. With more particles, this effect remains whenever parts of the data closely but imperfectly mimic the target, as is the case in the sequences we analyze in this thesis.

3.2 Dynamics of the Basic Particle Filter

The specification of the dynamical distribution $P(\mathbf{X}_t|\mathbf{X}_{t-1})$ is important to the successful performance of a particle filter. For a particle filter to track correctly and in the allotted processing time, it is important that the samples which are measured are drawn from an optimal location in the state space, and the accurate determination of the state dynamics is one way to optimize the location of these samples.

Object dynamics are sometimes represented a second order auto-regressive process, with additive noise. Therefore,

$$\mathbf{X}_t = A_2\mathbf{X}_{t-2} + A_1\mathbf{X}_{t-1} + \mathbf{D}_0 + B_0\mathbf{w}_t$$

where $\mathbf{w}_t$ are independent vectors of independent standard normal variables. A_1, A_2 are matrices representing the deterministic components of the dynamical model, B_0 represents

the stochastic component, and $\mathbf{D}_0$ is a steady state dynamical term. We may also express this by forming a vector

$$\mathcal{X}_t = \begin{pmatrix} \mathbf{X}_{t-1} \\ \mathbf{X}_t \end{pmatrix},$$

so that

$$\mathcal{X}_t = A\mathcal{X}_{t-1} + \mathbf{D} + B\mathbf{w}_t,$$

where

$$A = \begin{pmatrix} 0 & I \\ A_2 & A_1 \end{pmatrix}, \quad \mathbf{D} = \begin{pmatrix} 0 \\ \mathbf{D}_0 \end{pmatrix} \text{ and } B = \begin{pmatrix} 0 \\ B_0 \end{pmatrix}.$$

3.2.1 Specifying the dynamical variables

With the dynamics defined this way, the system becomes a set of damped oscillators, with natural frequencies and damping constants determined by A, and with external additive noise determined by $\mathbf{B}$. To set these constants by hand in the case of a single stochastic variable x_t where

$$x_t = a_2 x_{t-2} + a_1 x_{t-1} + b w_t,$$

we can assert our damping constant β, our natural frequency f, our root mean square average displacement ρ, with a system time step of length τ. We would then have:

$$a_1 = 2\exp(-\beta\tau)\cos(2\pi f\tau)$$

$$a_2 = -\exp(-2\beta\tau)$$

and

$$b = \rho\sqrt{1 - a_2^2 - a_1^2 - 2\frac{a_1^2 a_2}{1 - a_2}}.$$

This mode of specifying the dynamical parameters is generalizable to multiple state variables.

If dynamics are taken into consideration, this will typically be done by including an additional dimension representing the velocities of all its current dimensions. Stochastic diffusion noise may be added to the components of the state vector (which usually represent position) of the particle after it has been operated on by the transition matrix A, and noise corresponding to acceleration in the particle may be then added to the velocity dimension of the corresponding position dimension.

So, with x, y, z representing 3D coordinates for example,

$$\mathbf{X}_t = (x, y, z)$$

would be replaced with

$$\mathbf{X}_t = (x, y, z, \dot{x}, \dot{y}, \dot{z}),$$

and in this case

$$A = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

Implementations of particle filtering usually do not include any dynamics other than stochastic diffusion. The extra dimensions required in the particle vector essentially double its size, and although there is a high degree of dependence between each velocity element and its corresponding position element, the number of samples taken would usually have to increase to compensate for the larger size of the state space, resulting in a higher computational load. In addition, in the target tracking domain, it is often the case that the velocity of the object changes too suddenly and too frequently for the inclusion of a velocity term to be expedient.

In the case where the state space represents something more complicated than simple position and orientation, but rather something like a configuration in shape-space, a dynamical transition matrix may not be obviously specified. In such a case, is may be useful to learn a dynamical model from typical training data. This method is in fact also applicable to the case of a position/orientation state space, and would allow us to use a dynamical model without including a velocity term in the particle vector, but in target tracking problems, it is not usual to be able to infer useful general dynamical trends except for specific problems, e.g. people (targets) generally walking in a particular direction and then generally turning in a particular direction when they have reached a particular point.

3.2.2 Learning a dynamical model

The following algorithm, taken from [7], allows us to calculate estimates for the matrices which represent the state space dynamics.

Given a correct sequence of points in state space $(\mathbf{X}_1, \mathbf{X}_2, .., \mathbf{X}_N)$,

1. Sums and autocorrelations are calculated:

$$\mathbf{K}_i = \sum_{k=3}^{N} \mathbf{X}_{k-i}, \quad K_{ij} = \sum_{k=3}^{N} \mathbf{X}_{k-i}\mathbf{X}_{k-j}^T, \quad K'_{ij} = K_{ij} - \frac{1}{N-2} K_i K_j^T$$

2. Then we may calculate the matrices for the dynamics using

$$A_1 = (K'_{01} - A_2 K'_{21}) K'^{-1}_{11})$$

$$A_2 = (K'_{02} - K'_{01} K'^{-1}_{11} K'_{12})(K'_{22} - K'_{21} K'^{-1}_{11} K'_{12})^{-1}$$

$$\mathbf{D}_0 = \frac{1}{N-2}(\mathbf{K}_0 - A_2 \mathbf{K}_2 - A_1 \mathbf{K}_1)$$

3. Then B_0 is the matrix square root $B_0 = \sqrt{C}$ with

$$C = \frac{1}{N-2}(K_{00} - A_2 K_{20} - A_1 K_{10} - \mathbf{D}_0 \mathbf{K}_0^T)$$

Chapter 4

Measuring conditional probability of observations: Gaussian Mixture Models and Colour Spaces

We have seen in the previous chapter how to implement the sampling and dynamics stages of the particle filtering algorithm, what remains to understand the basic particle filter is to describe the method of taking the observation for a particular sample, i.e. how to calculate $p(\mathbf{Z}_t|\mathbf{X}_t = \mathbf{s}_t^{(i)})$. The observation of a particle seeks to calculate the likelihood that the hypothesis has caused the current observed data. Within the problem domain of particle filtering for tracking people, we may say that in general there have been two approaches: contour based, and colour/histogram based. Each has its own advantages. The contour based approach has the benefit of allowing greater detail to be used in the hypothesis model, since the locations of body parts (limbs, torso) may be detected and independently verified via an edge detection matching against an image which has been preprocessed by an edge detection filter, such as the Canny, Sobel or other.

The benefit of using a colour/histogram measure is that very often the colour properties of a target object vary very little with the aspect of the object, so that a total object localization becomes easy to do, although perhaps the particular configuration of the object would be more difficult to calculate. If the colour information known about the target is not useful in distinguishing it in the scene, then one may need to rely on contour, or other information. However, for this implementation, where the environment is controlled, and the targets are wearing identifiable colours, we prefer using a colour based method. In particular we seek to classify the image data on a per pixel level, using the available colour descriptions of the targets. We therefore use Gaussian Mixture Models, which have been shown many times to be highly applicable to this class of problem. The method with which

the GMMs are trained is based on the Expectation Maximization algorithm, and is due to Nabney, [67]. The reader may refer to Appendix A for a full description of this algorithm.

4.1 Colour spaces

The usual Red-Green-Blue (RGB) representation of colour data is just one of many different available colour spaces. In fact, there are many reasons why the RGB space is less useful than other spaces, and for this reason we need to examine the use of other colour spaces for the representation of our colour data, and for the formation of the Gaussian Mixture Models for out target objects. Different spaces yield different models, as can be inferred upon examination of colour space scatter plots of data points (pixel values) in sample distributions taken from the various target objects. There are a number of different colour spaces at our disposal. The effect of using different colour spaces, given that our original data is recorded in RGB space, will be to adjust linearly or nonlinearly the location of each colour point, as well as the distances between colours. Thus our methods for comparing colour distributions will be affected, as will the calculation of the probability of a pixel of a particular colour belonging to a particular mixture model.

4.1.1 RGB

Human beings have three types of photoreceptors, which are sensitive, approximately, to red, green and blue colour wavelengths. When we want to capture any particular colour, we may do it with colour detection sensors in such a way that enough information is stored to stimulate human photoreceptors to experience the same colour on reproduction. One way of doing this is to use sensors which have approximately the same frequency response as each human photoreceptor.

For any colour capture device, we define the red, green and blue components of a particular light as

$$R = \int_{300nm}^{830nm} S(\lambda)R(\lambda)d\lambda,$$

$$G = \int_{300nm}^{830nm} S(\lambda)G(\lambda)d\lambda,$$

$$B = \int_{300nm}^{830nm} S(\lambda)B(\lambda)d\lambda,$$

where $R(\lambda)$, $G(\lambda)$, $B(\lambda)$ are the red, green and blue sensors' respective sensitivity to wavelength λ, and $S(\lambda)$ is the incoming light spectrum. Since the definitions of red, green

and blue depend on the capturing device, we notice that the RGB space is therefore itself a device-dependent colour space. There exist methods for calibrating any device-dependent RGB data into perceptually uniform colour spaces, and providing this is done, we may subsequently transform the calibrated data into other spaces. One reason why we may wish to perform such transformations is that in the RGB space, there is high correlation between the R,G and B components in natural images. Another reason is the perceptual non-uniformity in the RGB space, where there is a low correlation between the Euclidean distance between two colours and their perceived colour difference.

4.1.2 Opponent colour space

This colour space was constructed by Ewald Hering after he noted that certain hues never appear together, for example red-green or yellow-blue. This provides a natural way to formulate a colour space which had low correlation between components in natural light. It was later discovered that there is a layer in the human visual system which converts the RGB stimuli of the cones into an opponent colour space. The new components are:

$$RG = R - G,$$
$$YeB = 2B - R - G,$$
$$WhBl = R + G + B.$$

There is also a logarithmic opponent colour transformation:

$$RG = \log R - \log G,$$
$$YeB = \log B - \frac{(\log R + \log G)}{2},$$
$$WhBl = \log G.$$

4.1.3 Ohta transformation

This is an approximation to the Karhunen-Loeve transformation of the RGB components of natural colours, and was developed for a region segmentation application by Ohta, Kanade and Sakai in 1980 [70]. The components are very well decorrelated:

$$I_1 = \frac{R+G+B}{3},$$
$$I_2 = \frac{R-B}{2},$$
$$I_3 = \frac{2G-R-B}{4}.$$

4.1.4 YUV and YIQ

The YUV and YIQ colour spaces are used in the transmission of analogue television, the former in Europe, the latter in North America. These spaces seek to encode the luminance in the Y component, and the chromaticity information in the other two channels. All three channels are orthogonal, and relatively decorrelated for naturally occurring colours. These colour spaces are well known, and widely used.

$$\begin{bmatrix} Y \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.257 & 0.504 & 0.098 \\ -0.148 & -0.291 & 0.439 \\ 0.439 & -0.368 & -0.071 \end{bmatrix} \times \begin{bmatrix} R \\ G \\ B \end{bmatrix} . + \begin{bmatrix} 16 \\ 128 \\ 128 \end{bmatrix}$$

4.1.5 HSV

Hue-Saturation-Value is one of the so called phenomenal colour spaces, where a colour is described by its hue (redness, yellowness, etc), saturation (level of non-whiteness), and value (total brightness of perceived colour). This is the mind's most natural way of classifying colour. There are a number of different definitions for the transformation from RGB to HSV space. One such definition is:

$$H = \begin{cases} \frac{G-B}{\max(R,G,B)-\min(R,G,B)} & \text{if} \quad \max(R,G,B) = R \\ 2 + \frac{B-R}{\max(R,G,B)-\min(R,G,B)} & \text{if} \quad \max(R,G,B) = G \\ 4 + \frac{R-G}{\max(R,G,B)-\min(R,G,B)} & \text{if} \quad \max(R,G,B) = B \end{cases}$$

$$S = \frac{\max(R,G,B) - \min(R,G,B)}{\max(R,G,B)},$$

$$V = \max(R,G,B).$$

If the saturation is zero, then the hue is undefined.

4.1.6 XYZ

The XYZ transformation, proposed by the International Commission on Illumination (CIE) along with L*a*b*, is used as a linear transformation on the RGB data before the L*a*b* conversion is applied. Not all visible colours may be represented by linear combinations of primary colours in RGB space. To solve this, we may use XYZ space, in which all visible

colours may be represented as linear combinations of X,Y and Z, which are not real colours, and are so termed virtual primaries. The transformation is:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.412453 & 0.357580 & 0.180423 \\ 0.212671 & 0.715160 & 0.072169 \\ 0.019334 & 0.119193 & 0.950227 \end{bmatrix} \times \begin{bmatrix} R \\ G \\ B \end{bmatrix}.$$

4.1.7 L*a*b*

The L*a*b* is a nonlinear transformation from XYZ space, and usually RGB data is first transformed into XYZ, then to L*a*b*. It was proposed in 1976 by the CIE, and was intended to be a perceptually uniform colour space [69], suitable for measuring colour differences under daylight or similar conditions. The XYZ values are normalized by the so called white point, and this allows us to use the L*a*b* space in different lighting conditions.

$$L* = \begin{cases} 116(\frac{Y}{Y_n}^{\frac{1}{3}}) - 16 & \text{if} \quad \frac{Y}{Y_n} > 0.008856 \\ 903.3(\frac{Y}{Y_n} - 16) & \text{if} \quad \frac{Y}{Y_n} \leq 0.008856 \end{cases}$$

$$a* = 500[f(\frac{X}{X_n}) - f(\frac{Y}{Y_n})]$$

$$b* = 200[f(\frac{Y}{Y_n}) - f(\frac{Z}{Z_n})]$$

where

$$f(t) = \begin{cases} t^{\frac{1}{3}} & \text{if} \quad t > 0.008856 \\ 7.787 \times t + \frac{16}{116} & \text{if} \quad t \leq 0.008856. \end{cases}$$

In the L*a*b* space, perceptual colour differences can be quantified as the Euclidean distance between points in this space.

4.2 Extracting GMM models for target objects

Models for the target objects may be obtained by taking one or more manually segmented frames, possibly from the tracking sequence itself, and extracting a set of RGB points for each target object. These data points are then used to create a Gaussian Mixture Model for each target object.

A set of Gaussian Mixture Models is created in this way for each target object, with the RGB data converted to various colour spaces beforehand. Although this may make a difference for example in the positions of the means and in the final covariance matrices for each model, we do not investigate the effects of different colour spaces on the tracking algorithm itself. Instead, we inspect the probabilistic pixel labelling produced by comparing the pixels of an image with respect to each of the GMMs we have created, and investigate a metric which we have developed to test the correctness of the pixel classification in these probability maps. The probability maps presented in section 4.6 demonstrate the quality of the probabilistic pixel labelling produced by this process.

4.3 Developing a set of GMMs for the background

Assuming that the camera and scene are static, we have at our disposal information about what the static background looks like. We can use this to assist in the probabilistic classification of individual pixels in the scene.

The approach taken in this implementation with respect to the classification of the individual pixels, is to develop a GMM for each background pixel according to its behaviour as observed over several frames. Since the camera and the scene are largely static, it was deemed sufficient to approximate the behaviour using a single Gaussian center and covariance matrix, although the framework allows for individual pixel models with an arbitrary number of mixture centers. An alternative approach, used in [1], is to create GMMs for a region of pixels. This would assist in situations where parts of the background actually move, such as the leaves of trees blowing in wind. However, since we are dealing with an indoor intelligent environment, we do not expect the scene to behave in this way.

When calculating the probability that a pixel belongs to a particular model, we create the activation matrix **A**, as we did when training the GMM for each object model (see Appendix A), using all the available GMM object models and their associated means and covariances. **A** contains in its entries probabilities of data points being caused by each mixture center. From this we create a normalized posterior matrix **R**, which is a normalized causality matrix which also takes into account the prior probabilities of the mixtures centers in **A**. Included in this set of GMMs is the GMM for the background model for the pixel which is currently being tested.

Care must be taken to label each mixture component with the model to which it belongs. Our new vector of mixture center priors, which originally looked like

$$\mathbf{P} = \begin{bmatrix} p_1 & p_2 & \cdots & p_K \end{bmatrix},$$

will now be composed of labelled entries,

$$\mathbf{P} = \begin{bmatrix} p_1^1 & p_2^1 & p_3^2 & p_4^2 & p_5^3 & p_6^3 & \cdots & p_K^5 \end{bmatrix},$$

where p_n^m indicates that this element of the prior vector $\mathbf{P}$ is associated to mixture component n, which belongs to model m, where m is an element of the set of h available mixture models. Typically we will have one mixture model per target object, and one for the background at the current pixel.

It is more convenient for the notation to represent the models to which each mixture center belong in a separate vector:

$$\mathbf{L} = \begin{bmatrix} 1 & 1 & 2 & 2 & 3 & 3 & \ldots & 5 \end{bmatrix}.$$

Assuming that there is no a priori bias towards any one of the models, we need to normalize the prior vector in such a way that the sum of prior probabilities for each model are equal:

$$\sum_{i=1}^{K} \delta_m(\mathbf{L}_i) p_i = \frac{1}{h} \qquad m = 1..h$$

where h is the number of models we are using, including the background model, and $\delta_m(.)$ is a dirac delta centered at m. The number of mixture centers per model may thus vary if desired.

4.4 Samples representing locations of target objects

In this implementation, we seek to track moving targets, which are people, through a video sequence. The state space in which our state distributions exist represents the joint space of all the 3D locations of the target objects. For a single person, our state vector would consist of that person's x,y and z position, in world coordinates. Since a single particle in our prior, dynamical or posterior distribution represents the location of all the people in the scene, the particle vector is a concatenation of the x,y and z coordinates of all of the people in the scene. If there are N people being tracked in the scene, then the j_{th} particle at time t might look like

$$\mathbf{s}_t^{(j)} = [x_1, y_1, z_1, x_2, y_2, z_2, ..., x_N, y_N, z_N]^T,$$

Figure 4.1: Image projection of sample generated at a particular time in a particle filter, or partitioned particle filter, in this implementation.

if there are N people in the scene.

To take an observation of an object using image data, we need to project the location of the hypothesized region in which the person lies into image coordinates. In the spirit of [1], we choose to represent the region which a person occupies as an ellipse in world coordinates. This 3D conic benefits from many invariant properties as well as being simple to render into a calibrated scene. In this implementation we use the OpenGL rendering pipeline which fits neatly into the C++ code in which the rest of the particle filtering algorithm is written.

All the ellipsoids are rendered into an empty scene, which we may then search through to discover the pixel labelling generated by any sample hypothesis. The use of a joint hypothesis of all target objects within a single particle has the added advantage that mutual occlusion between the target objects may be modelled in terms of the pixel labelling after the rendering stage. As examples of joint hypothesis ellipsoids generated by particles, we see Fig. 4.1, Fig. 4.2 and Fig. 4.3.

4.5 Observing particles using Gaussian Mixture Models

Now that we have developed a method for assessing the probability that any pixel belongs to any of a set of models, we are equipped to formulate the method by which we take the observation $p(\mathbf{Z}_t|\mathbf{X}_t = \mathbf{s}_t^{(j)})$ of a particular sample $\mathbf{s}_t^{(j)}$. Given that each particle is projected as an ellipse into the scene, and that these ellipses may be regarded as hypothesized pixel

Figure 4.2: Image projection of sample generated at a particular time in a particle filter, or partitioned particle filter, in this implementation.

Figure 4.3: Image projection of sample generated at a particular time in a particle filter, or partitioned particle filter, in this implementation.

Figure 4.4: The image projection of such a hypothesis will be compared to the data image. If the hypothesis is good, then the target objects which the ellipsoids represent will occupy the same region in image space as the people.

labellings for the current frame, the observation $p(\mathbf{Z}_t|\mathbf{X}_t)$ of a particle may be calculated as

$$p(\mathbf{Z}_t|\mathbf{X}_t) = \prod_{g=1}^{G} p(z_g|\mathbf{X}_t) = \prod_{g=1}^{G} p(z_g|l_g),$$

where g is the index of a particular pixel on the image grid, which contains G pixels in total. The labelling of pixel g as hypothesized by the current particle $\mathbf{s}_t^{(n)}$ is l_g where $l_g \in \{1, 2..h\}$ with h the number of models/target objects, including the background. We may see an example of how the ellipsoids are designed to represent target objects in Fig. 4.4, where the ellipsoids are superimposed on the corresponding frame in the sequence. After all particles have been observed, the Maximum a Posteriori (MAP) estimate is found. Fig. 4.5 shows the MAP estimate for a particular frame in the indoor sequence.

Note that here independence between all pixels with respect to their observations given the hypothesized labelling of that pixel is assumed. It is possible here to improve the probability estimate via a region based homogeneity measure, possibly via Markov Random Fields, to take advantage of the fact that pixels of a target object are usually adjacent to one another, and thereby discarding the aforementioned simplifying independence assumption. After we have calculated an activation matrix based on the current pixel as the data, given our combined Gaussian Mixture Model, we calculate for the combined GMM a normalized $1 \times K$ posterior matrix $\mathbf{R}$. This step is otherwise identical to the calculation of the posterior

Figure 4.5: After each of the partitions has been processed, we are interested in the Maximum a Posteriori estimate of the state space. The sample with the highest probability is stored and displayed.

matrix $\mathbf{R}$ in the training of the GMM, described in Appendix A, although $N = 1$, since there is only one data point, i.e. the current pixel.

Again we develop a likelihood for each of the mixture centers:

$$p_j^l = \mathbf{R}_{0,j} \qquad \{j = 1..K\}.$$

The probability $p(z_g|l_g)$ is then

$$p(z_g|l_g) = \sum_{i=1}^{K} \delta_{l_g}(\mathbf{L}_i) p_i^l \qquad \{m = 1, 2, .., h\}$$

where h is once again the number of mixtures models in the GMM, and $\mathbf{L}$ is the labelling vector for the GMM mixture centers.

For notational convenience, we may rewrite

$$p(z_g\, l_g) = p_{l_g}(z_g)$$

As pointed out in [1] and [53], with the pixel probabilities defined this way, and since the observation density only needs to be evaluated up to a multiplicative constant, we can write

$$p(\mathbf{Z}_t|\mathbf{X}_t) = \prod_{g=1}^{G} p(z_g|\mathbf{X}_t) = \prod_{g=1}^{G} p_{l_g}(z_g) = \prod_{g \in F} p_{l_g}(z_g) \times \prod_{g \in B} p_B(z_g)$$

where $p_B(.) = p_{l_g}(.)$ with $l_g = B$, F is the set of all pixels which are hypothesized to lie in the foreground, and B is the set of hypothesized background pixels.

Using this observation, we may normalize the expression for the observation:

$$p(\mathbf{Z}_t|\mathbf{X}_t) \propto \frac{\prod_{g\in F} p_{l_g}(z_g) \times \prod_{g\in B} p_B(z_g)}{\prod_{g=1}^{G} p_B(z_g)} = \frac{\prod_{g\in F} p_{l_g}(z_g)}{\prod_{g\in F} p_B(z_g)} = \prod_{g\in F} \frac{p_{l_g}(z_g)}{p_B(z_g)}.$$

This allows us to visit only the foreground pixels when we perform our observation using the image data.

As also indicated in [1] and [53], another useful formulation is to convert all the probabilities in an image-preprocessing stage into the log domain. Then we will have:

$$\log(p(\mathbf{Z}_t|\mathbf{X}_t)) = \log\left(\prod_{g=1}^{G} p(z_g|\mathbf{X}_t)\right) = \sum_{g=1}^{G} \log\left(p(z_g|\mathbf{X}_t)\right).$$

This allows one to replace the product term with a sum term, which often speeds up the processing as well as removing the underflow problem, which is arises due to the large number of pixels in an image.

4.5.1 Underflow

A problem which is solved by conversion into the log-domain is that if the probabilities are simply calculated according to

$$p(\mathbf{Z}_t|\mathbf{X}_t) = \prod_{g=1}^{G} p(z_g|\mathbf{X}_t) = \prod_{g=1}^{G} p_{l_g}(z_g),$$

this results in an extremely small value for $p(\mathbf{Z}_t|\mathbf{X}_t)$, often well below machine precision. If the pixel classification stage for a correct hypothesis attributes to each pixel a 0.9 probability of belonging to the object which it has been hypothesized as belonging to (i.e. a very good hypothesis), then the fact that there are approximately a hundred thousand pixels will cause the probability to be calculated as

$$p(\mathbf{Z}_t|\mathbf{X}_t) = \prod_{g=1}^{G} 0.9 = 0.9^G,$$

which is approximately 10^{-4500}. When working in the log domain, all that needs to be stored is the exponent, which is within the range of a four byte floating point variable.

4.5.2 A lower bound for pixel classification probabilities

It is important also to set a lower bound on the classification of any pixel with respect to any model. Since the observation process is essentially a multiplicative process where particles are scored according to their relative merit, if any single pixel contributes a zero probability factor to the product, this will set the probability of the particle to zero, although all of its other pixel classification hypotheses may have been very good. In this implementation, we set a lower bound of $\exp(-100)$ to all the pixel classifications which have associated probabilities of $\exp(-100)$ or lower.

4.6 Implementation results

We have experimented with the use of some of the different colour spaces, and the colour separation which we get by using them. Also we have investigated the resulting effects of performing pixel classification according to the Gaussian Mixture Models which we form from data from these different colour spaces. Before each image is processed, the image data is converted to the appropriate colour space for pixel classification.

4.6.1 Scatter plots in different colour spaces

First we present, for the particular tracking task at hand (i.e. that of tracking people wearing highly coloured clothing), histogram plots of the different people. From these histograms may be seen the colour separation already present in the RGB information, and the improvements gained when using different colour spaces. In each of the histograms shown in Fig. 4.6, Fig. 4.7, Fig. 4.8 and Fig. 4.9 each data point belongs to one pixel taken from a set of training images, and this is the same data which was used to generate the GMMs. There is saturation in the RGB data of the colour histograms of certain of the targets, which may be seen most clearly in Fig. 4.6. It is impossible to say whether the classification would have been significantly better if this saturation had not been present. However, since this saturation is present in the RGB data, from which we transform the data to the various other colour spaces, the effect of the saturation is present in all of the GMMs trained in each of the colour spaces.

YUV space seeks to decorrelate the components on the assumption that one is dealing with naturally occurring colours. Using YUV we see in Fig. 4.7 a slightly better colour separation than in the RGB histogram in Fig. 4.6.

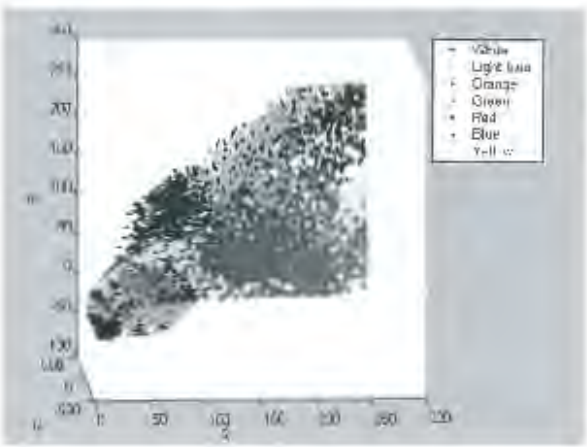

Figure 4.6: Scatter plot of colour data of people in indoor scene in RGB space. There are seven people, labelled "White", "LightBlue", "Orange", "Green", "Yellow", "DarkBlue" and "Red".

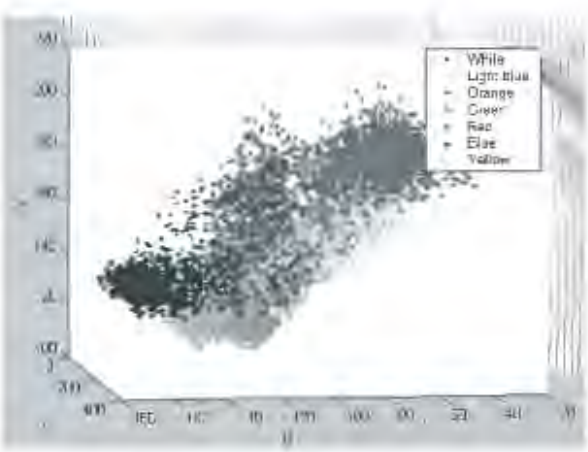

Figure 4.7: Scatter plot of colour data of people in indoor scene in YUV space

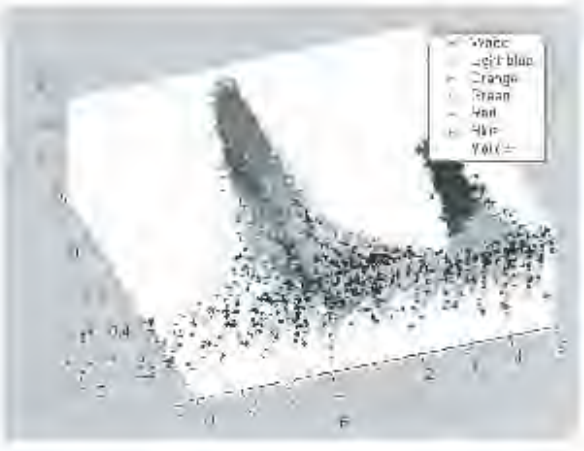

Figure 4.8. Scatter plot of colour data of people for indoor scene in HSV space

The HSV based pixel classification appears to yield poor colour separation, as can be seen in Fig. 4.8.

The L*a*b* transformation appears to have the best colour separation (Fig. 4.9), and was the one selected to generate the GMMs for this implementation.

4.6.2 Probability maps using different colour spaces

We have generated probability maps, which demonstrate the variability in the pixel classification generated for a specific frame, using the Gaussian Mixture Models generated from the data in the different colour spaces discussed above. The GMMs were trained in the same way given the data in each of the different colour spaces, and each GMM has sixteen mixture centers. The choice of sixteen centers per GMM per target may be said to be excessive given the uniformity of the colour distributions of each of the individual targets. The number was chosen for two reasons. Firstly, the distributions are less uniform than they appear, given the varying aspects of each person to the light sources in the sequence. Also, the skin colours of some of the targets vary, and this is relevant information for the tracker. Secondly, the outdoor sequences, which are also tested to a degree, require sixteen centers per target (as has been established empirically), and the consistency of the number of centers per model across sequences allows better comparison of other aspects of the tracker. In Fig. 4.10 the correct pixel labelling for a frame in the benchmark scene is displayed, and the probabilistic classifications shown in Fig. 4.11, Fig. 4.12, Fig. 4.13

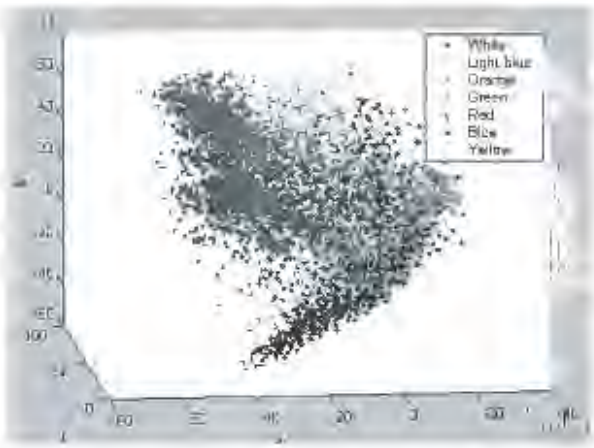

Figure 4.9: Scatter plot of colour data of people for indoor scene in L*a*b* space.

and Fig. 4.14 may be compared with this data. In each image, the likelihood of a pixel belonging to the specified target object is used to generate a proportional pixel intensity on the grey scale. Therefore a white pixel represents a near certainty that it belongs to the specified object, and a black pixel represents a near zero probability.

We can see the classification of the white person in Fig. 4.11, using RGB data, is not particularly good.

The GMMs trained in YUV space, shown in 4.12 yield good results. The GMMs trained using the HSV data, shown in Fig. 4.13, yield poor results. This is because of the poor clustering which we observed in the histogram plot in the training data (this result is corroborated in [9]).

The results on GMM classification using training data in the L*a*b* colour space is visibly superior, and empirical results show why we should use this space for our implementation of the particle filter based target tracker.

4.6.3 Ground-truth comparison of pixel classification

Since we have at our disposal manually labelled data for each of the frames in the sequence, it is useful to compare the pixel classifications as generated by the different GMMs with this manually labelled data. The following metric, introduced here and called the "Correct Classification Metric", C (which may also be written as $C_{\text{classification method}}$, where *classification method* indicates the method used for classification), may be used to assess the

Figure 4.10: Correct pixel classification for the 171st frame in the sequence

Figure 4.11: Pixel classification of the white person using GMMs trained with RGB data.

Figure 4.12: Pixel classification of the white person using GMMs trained with YUV data.

Figure 4.13: Pixel classification of the white person using GMMs trained with HSV data.

Figure 4.14: Pixel classification of the white person using GMMs trained with L*a*b* data.

accuracy of each of the GMM based pixel classifications. For example, with GMMs trained using the HSV colour space, we have:

$$C_{hsv} = \prod_{g=1}^{G} p_{hsv}(z_g|c_g) = \prod_{g \in T_0} p_{hsv}(z_g|c_g = T_0) \times \prod_{g \in T_1} p_{hsv}(z_g|c_g = T_1) \times \ldots \times \prod_{g \in T_N} p_{hsv}(z_g|c_g = T_N),$$

where $p(z_g|c_g = T_i)$ is the probability of observing pixel data z_g, given that pixel g has been correctly (manually) labelled as belonging to target T_i, $g \in T_i$ are the pixels g in the region of the image which has been labelled as belonging to target T_i, and where N is the number of different models including the background. The probability function $p_{hsv}(.)$ indicates that the probability will be calculated using GMMs trained in HSV space.

Introducing the Log Correct Classification Metric

$$K_{\text{classification method}} = -\log(C_{\text{classification method}}),$$

we may then say, in the HSV colour space:

	RGB	YUV	HSV	L*a*b*
$\mathcal{T}_0$	-1017.6	-726.2	-1462.8	-770.6
$\mathcal{T}_1$	-728.2	-494.8	-1096.7	-498.7
$\mathcal{T}_2$	-1614.3	-1097.5	-1905.9	-1120.2
$\mathcal{T}_3$	-1292.7	-464.2	-893.3	-481.8
$\mathcal{T}_4$	-1131.5	-1572.2	-2205.3	-1622.6
$\mathcal{T}_5$	-406.9	-412.6	-547.7	-428.8
$\mathcal{T}_6$	0	0	0	0
$\mathcal{T}_7$	-372109.5	-255010.5	-110228.0	-186916.6
K	-378301.1	-259778.1	-118339.7	-191839.5

Table 4.1: Table of the classification quality for each of the target objects, in different colour spaces, using the Log Correct Classification Metric.

$$K_{hsv} = \log(C_{hsv}) = \log\left(\prod_{g=1}^{G} p_{hsv}(z_g|c_g)\right)$$
$$= \sum_{g\in\mathcal{T}_0} \log p_{hsv}(z_g|c_g = T_0) + \sum_{g\in\mathcal{T}_1} \log p_{hsv}(z_g|c_g = T_1) + \ldots + \sum_{g\in\mathcal{T}_N} \log p_{hsv}(z_g|c_g = T_N)$$
$$= \mathcal{T}_0 + \mathcal{T}_1 + \ldots + \mathcal{T}_N.$$

Using a manually labelled image in the sequence, we can generate the Table 7.1, which displays the target matching characteristics $\mathcal{T}_i$ for each of the colour spaces, for the classification of the same frame which the probability maps of Fig. 4.11, Fig. 4.12, Fig. 4.13 and Fig. 4.14 represent. Table 7.1 contains some interesting information about the accuracy of the classification which we can expect when using GMMs trained on data from different colour spaces. What we first see is that the value for K, which is the last row in Table 7.1, is the highest for the HSV colour space, which indicates that this is the colour space we should use. However, when we inspect the individual terms $\mathcal{T}_i$ for each of the target objects (the sixth one of which is absent from the scene), we see that the HSV colour space scores the worst in this regard (it is the most negative). The summation for the seventh target object, $\mathcal{T}_7$, represents the contribution of the term for the correct classification for all background pixels. The nature of the algorithm is such that the accurate classification of background pixels as being background pixels is not nearly as important as the accurate classification of foreground pixels belonging to the correct target. To put it differently, a high value for $\mathcal{T}_i$ in a particular colour space indicates that that target was well classified, using GMMs trained on data from that colour space. A low value for $\mathcal{T}_i$ indicates that the

Figure 4.15: Image from simple sequence with naturally coloured people.

target was poorly classified. Therefore, as a compromise between these two desirable traits in a pixel classifier, we use the GMMs trained on data in L*a*b* space.

4.6.4 Probability maps for scene with naturally coloured targets

For completeness, and for comparison with the benchmark indoor sequence, we examine also the colour histograms generated with the manual segmentation of people from a natural scene, i.e. a scene in which the motion is simpler than the indoor benchmark video sequence used for this thesis.

The colour histograms taken from manual segmentation of the people in this scene are depicted in Fig. 4.16 and Fig. 4.17, in which we can see that in natural scenes, the colour histograms of people overlap far more, and this will lead to an inferior, although usable, probability measure in the observation stage for the particle filter. We also examine the probability maps generated from an image in this natural sequence, to see if the pixel classification is accurate enough to be used as a basis for a particle filter tracking implementation. We can see from the two probability maps in Fig. 4.19 and Fig. 4.20, which may be compared with manually classified data in Fig. 4.18, that the pixel classification becomes more difficult when the people are wearing natural, random clothing. This is reflected also in the histograms shown in Fig. 4.16 and in Fig. 4.17 where there is lots of overlap between the scatter plots of the different targets. The usefulness of colour models in identifying targets is only as good as the colour separation in the colour histograms of the targets involved.

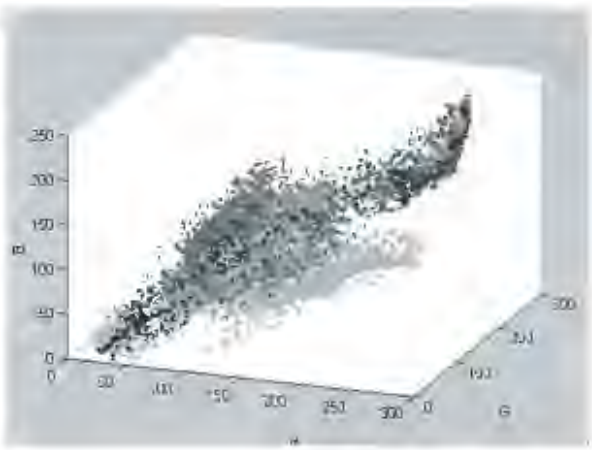

Figure 4.16: Scatter plot of colour data of four people in natural scene in RGB space.

We may see for example that there is considerable confusion between two of the targets in Fig. 4.19, as is indicated by the two different targets appearing as white regions in the same probability map. This arises because they are both wearing white T-shirts.

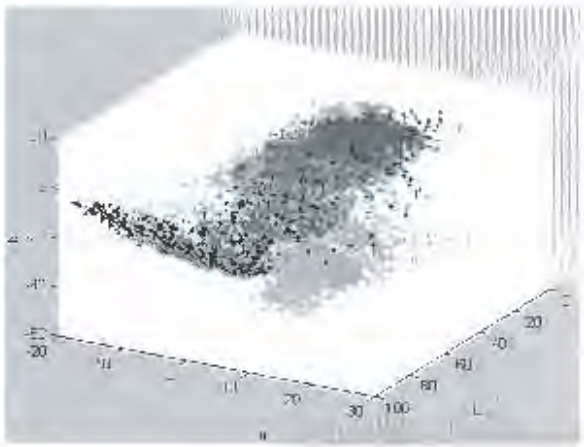

Figure 4.17: Scatter plot of colour data of four people in natural scene in L*a*b* space.

Figure 4.18: Manual pixel classification of the people in the 317th frame in the outdoor sequence.

Figure 4.19: Pixel classification of the first person using GMMs trained with L*a*b* data.

Figure 4.20: Pixel classification of the fourth person using GMMs trained with L*a*b* data.

Chapter 5

Hidden Markov models and fixed lag smoothing for refinement of pixel classification

To track an object moving through a scene, our task is to accurately label the pixels of each frame as having a correct probability of belonging to each of the target objects (for which we have generated Gaussian Mixture Models). Our GMMs are based on information we have about the colour distributions of the target objects in the scene, and of the background of the scene.

If we consider the nature of the movement of the target objects, we see that they tend to move slowly, and their position in any frame is highly dependent on their position in the previous frame. Therefore, given a pixel in a frame which belongs to a particular model, be it a model of one of the target objects or of the background, it is likely that in the next frame it will belong to the same model. If we then create a Hidden Markov Model for each pixel, and associate with each model a state in the pixel's HMM. Then, given a range of observations of that pixel over a series of frames, we can impose our knowledge of the restrictions described earlier on the pixels state transition dynamics, via the transition matrix of the pixel's HMM. This approach is taken in [20], although in that implementation each pixel has five associated states (background, leading edge, target centre region, trailing edge, and colour change), which are used for all of the targets together. In contrast, this implementation has one pixel state per target.

5.1 Hidden Markov Models

5.1.1 Definition

A Hidden Markov Model (HMM) λ may be fully described by the 3-tuple $\lambda = \{\mathbf{A}, B, \pi\}$ (see [65] for a more in depth discussion). In this tuple, $\mathbf{A}$ represents the transition matrix, which contains the probabilities that the model, given that it is in a particular state i, will transition into another state j, B is a probability for each state j producing a certain observation, and π is a vector which represents the prior values on each of the states.

The HMM may be in one state q_t at time t of a set of states S, i.e. $s \in S$, with e.g. $S = S_1, S_2, S_3$

$$\mathbf{A}_{ij} = P[q_t = S_j | q_{t-1} = S_i]$$

with each state transition coefficient having the property that

$$\mathbf{A}_{ij} \geq 0$$

and

$$\sum_{j=1}^{N} \mathbf{A}_{ij} = 1$$

Also we require the definition of the observation probability distribution for each state. The formulation used in the case where each state's observation density is a Gaussian Mixture Model (GMM), is:

$$b_j(O) = \sum_{m=1}^{M} p_{jm} N(O | \mu_{jm}, V_{jm})$$

where p_{jm} is the prior, μ_{jm} is the mean, and V_{jm} is the covariance on a mixture center m for the HMM state j.

Finally we require a prior distribution λ on the states in which the HMM may be in at the first time step.

5.1.2 The three problems of HMMs

There are three problems of interest which we must deal with if we are using HMMs, namely

- How to train the model parameters

- How to find the most likely state sequence given a set of observations $O_1O_2O_3..O_n$
- How to find the probability of a particular observation sequence given a HMM

We are most interested in the first two problems for this implementation.

The first problem corresponds to discovering the transition matrix A for the HMM of each pixel. Classically, all of an HMMs parameters may be optimized via the forward-backward algorithm, which is itself an Expectation Maximization algorithm (see Appendix A for explanation), but here the problem is simplified, since we have already developed in the previous chapter a method for evaluating the probability of any observation of the pixel at a frame given the GMMs of the target objects.

The second problem corresponds to the task of assigning a probability to a pixel's HMM to be in a particular state given the transition matrix A and the observed data over a sequence of frames. In this implementation, fifteen consecutive observations are used for each pixel and those fifteen observations are then used to calculate the smoothed probability estimate for that pixel at the time step which is at the midpoint in the fifteen observations. This is slightly clumsy, since we are lagging seven frames behind reality, but the results of this smoothing are of potential benefit.

5.2 Training the model

5.2.1 Training the Hidden Markov Models

The model was trained manually. Since all transitions are possible, although self transitions (i.e. transitions to the same state in which the model is already in) are more likely, we fill the square transition matrix with values along the diagonal columns with a relatively large probability, place a smaller probability on the elements which correspond to a transition from a state representing an object to the state representing the background, and divide the remainder equally among the other entries. This is because we expect a transition from an object to the background with greater probability than we expect a transition from an object to another object. For the final row, we expect a high self transition if the pixel is already in the background state, and transitions to all the other objects are equally distributed. This pixel state transition model is defined using the assumptions that the area in image space occupied by the people is relatively small compared to the area occupied by the background, and that the people move independently and that there is usually some space between them when they interact and occlude one another. This latter assumption causes us to realize that in general after a person walks through a particular pixel in image

space, they are more likely to reveal the background behind them than another person. In principle a transition matrix should be made up of the following entries:

$$\mathbf{A} = \begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & p_{33} \end{bmatrix},$$

in which each element p_{ab} indicates the probability of a transition from state a to state b. Many different transition matrices were tested and the results compared to ground truth, and the following is the matrix which produced the best results for the indoor sequence:

$$\mathbf{A} = \begin{bmatrix} 0.6154 & 0.0385 & 0.0385 & 0.0385 & 0.0385 & 0.0385 & 0.0385 & 0.1538 \\ 0.0385 & 0.6154 & 0.0385 & 0.0385 & 0.0385 & 0.0385 & 0.0385 & 0.1538 \\ 0.0385 & 0.0385 & 0.6154 & 0.0385 & 0.0385 & 0.0385 & 0.0385 & 0.1538 \\ 0.0385 & 0.0385 & 0.0385 & 0.6154 & 0.0385 & 0.0385 & 0.0385 & 0.1538 \\ 0.0385 & 0.0385 & 0.0385 & 0.0385 & 0.6154 & 0.0385 & 0.0385 & 0.1538 \\ 0.0385 & 0.0385 & 0.0385 & 0.0385 & 0.0385 & 0.6154 & 0.0385 & 0.1538 \\ 0.0385 & 0.0385 & 0.0385 & 0.0385 & 0.0385 & 0.0385 & 0.6154 & 0.1538 \\ 0.0435 & 0.0435 & 0.0435 & 0.0435 & 0.0435 & 0.0435 & 0.0435 & 0.6957 \end{bmatrix},$$

where the last row and column correspond to the state of the pixel belonging to the background. This particular matrix describes a scenario which corresponds to the type of sequence which this tracker was designed for, namely with a large number of target objects, but in which it is assumed the targets maintain a certain distance from each other, (as is normal behaviour). If this assumption is violated on occasion, there is no problem, but in general we seek to take advantage of this trend. The number of rows and columns of matrix **A** is obviously dependent on the number of target objects present in the scene, and the size and content of the transition matrix should change according to the number of target objects in the scene. The above matrix is then the one used when all seven of the target objects are simultaneously in the scene, although it is by no means optimal. Ideally we should like to train through data a unique HMM for each pixel. We might expect the transition matrices of pixels which are in the center of the room (lots of activity) to be very different from the transition matrices on the edges of the scene, where there is little activity. However, this is not explored further.

5.2.2 Finding the best observation sequence

There are two general ways to solve the problem of finding the most likely state sequence which generated a set of observations. The first finds the sequence of states which were individually most likely, irrespective of a definite decision having been made about the states before or after it in the state sequence at any point. Once again, the reader is referred to [65], on which this section is based.

Individually optimal states

To find individually optimal states within the sequence, we define

$$\gamma_t(i) = P(q_t = S_i|O, \lambda),$$

which can be reexpressed as

$$\gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{P(O|\lambda)} = \frac{\alpha_t(i)\beta_t(i)}{\sum_{i=1}^{N} \alpha_t(i)\beta_t(i)},$$

where $\alpha_t(i)$ is defined as

$$\alpha_1(i) = \pi_i b_i(O_1) \qquad \{i = 1..N\},$$

where $b_i(O_t)$ is the probability of the observation at time t being generated by state i. For $t > 1$, $\alpha_t(i)$ is defined recursively, and we form the so called alpha-trellis thus:

$$\alpha_t(j) = \left[\sum_{i=1}^{N} \alpha_t(i)\mathbf{A}_{ij}\right] b_j(O_{t+1}) \quad \{t = 1..T-1\} \quad \{j = 1..N\}.$$

Similarly for the beta trellis,

$$\beta_T(i) = 1 \qquad \{i = 1..N\},$$

and inductively

$$\beta_t(i) = \sum_{j=1}^{N} \mathbf{A}_{ij} b_j(O_{t+1}\beta_{t+1}(j)) \quad \{t = T-1..1\} \quad \{i = 1..N\}.$$

Then, for any time t, we can solve for the individually most likely state

$$q_t = \underset{\mathrm{i}}{\mathrm{argmax}} \ [\gamma_t(i)] \quad \{t = 1..T\} \quad \{i = 1..N\}.$$

The Viterbi algorithm

The second method is known as the Viterbi decoding algorithm, and tries to find the best overall hidden state sequence, given that the sequence must be viewed as a whole. This is often different to the concatenation of the individually optimal states.

The problem is to find the quantity

$$\delta_t(i) = \max_{q_1, q_2, \ldots, q_{t-1}} P[q_1 q_2 q_3 \ldots q_t = i, O_1 O_2 O_3 \ldots O_t | \lambda]$$

The Viterbi algorithm is similar to the calculation of the alpha trellis ($\alpha_t(i)$) of the previous subsection, with a maximization step instead of a summing step, and for details, the reader may refer to Appendix B

The output of the Viterbi algorithm is the most likely state sequence which explains the observed data over the time period, however, this is not necessarily useful to us. When we take the observation for a given particle sampled from a distribution, the method we use is to sum over the log-likelihoods of all the pixels in the image, with respect to the hypothesized pixel labelling generated by the particle. Now since the Viterbi algorithm returns only the optimal state sequence, we have no way of using this information in probabilistic terms. That is, we cannot use this information when we sum over the log likelihoods, since the Viterbi algorithm does not yield a set of probabilities for the states to which a pixel may belong, over which we can then sum the logarithms to calculate our posterior distribution.

This is why the Viterbi algorithm was rejected for use in this implementation in favour of the method of the previous subsection, viz. finding the individually optimal states and then with those the set of probabilities of any pixel being in any state at any time.

5.3 Results of using fixed lag HMM smoothing

Shown here are some probability maps generated when HMM smoothing is included in the GMM segmentation method described in the previous chapter. It appears that the HMM smoothing in general removes some of the effect of spurious pixel (probabilistic) misclassification, as well as improving the probability estimate within the area of each target object, including the background. The probability maps for the person dressed in white (Fig. 5.2, Fig. 5.4) and for the person dressed in light-blue (Fig. 5.3, Fig. 5.5) are shown (these are the two targets which are the most readily confused). The quality of this pixel classification may be compared to the manual classification of the targets, which is shown in Fig. 5.1.

Figure 5.1: Manually classified pixels for the 87th frame in the indoor sequence.

Figure 5.2: The basic pixel classification map for the first person as generated by Gaussian Mixture Models trained in L*a*b space.

Figure 5.3: The basic pixel classification map for the second person as generated by Gaussian Mixture Models trained in L*a*b space.

Figure 5.4: The HMM smoothed pixel classification map for the first person.

Figure 5.5: The HMM smoothed pixel classification map for the second person.

When one examines the Log Correct Classification Metric, introduced in the previous chapter, we see some surprising empirical results based on these probability maps and the manual segmentation.

The Log Correct Classification Metric is worse for all targets, and for the total, $K_{\text{classification method}}$, which is the same metric, for any particular classification method (be it a GMM based classification method, GMM with HMM, or otherwise). This is a surprising result, since the probability maps seem to demonstrate an improvement in the classification. This empirical worsening of the probabilistic classification in the pixels is the result of the nature of the HMM smoothing, which tends to force the probability of classification of a pixel to a model to be closer to zero or closer to unity than the simple GMM based classification. After the HMM smoothing stage, we find in general that pixels which have a high probability in their classification of belonging to the correct (manually labelled) model, have had their probabilities for this classification increased further, and correctly so. However, in the case of pixels which are probabilistically misclassified, the probability of the classification to the correct model is decreased further, closer to zero. The overall effect is then to decrease the Log Correct Classification Metric for the pixel classification for a particular image. The total effect is negative as can be seen in the K parameter in Table 5.1.

There is however a beneficial aspect to the HMM smoothing, and this is that pixels which have been mistakenly classified with high probability as belonging to a target, which do not in fact belong to that target, will in general have their probabilities adjusted to

	L*a*b*	L*a*b* with HMM
T_0	-648.7	-778.3
T_1	-1325.0	-2157.5
T_2	-629.3	-1159.2
T_3	-481.3	-900.0
T_4	-199.7	-325.4
T_5	-484.6	-766.0
T_6	-377.1	-403.1
T_7	-243520.6	-381800.6
$K_{\text{classmethod}}$	-247666.7	-388290.5

Table 5.1: Log Correct Classification metric for each of the targets in the scene, at a typical frame. The addition of HMM smoothing appears to have worsened the pixel classification.

reflect more accurately their true targets, by the HMM smoothing step. This effect may be seen by the Log Erroneous Classification Metric, here introduced, which is in a sense the inverse of the Log Correct Classification Metric. We define the Erroneous Classification Metric E_{T_N} for a particular target T_N as

$$E_{T_N} = \prod_{g \in \overline{T_N}} p(z_g | c_g = T_N),$$

and similarly, the Log Erroneous Classification Metric is

$$S_{T_N} = \log(E_{T_N}) = \log \prod_{g \in \overline{T_N}} p(z_g | c_g = T_N) = \sum_{g \in \overline{T_N}} \log p(z_g | c_g = T_N),$$

which will have a large value if there have been many pixels erroneously classified with large probability as belonging to target N, and small if there were few pixels incorrectly labelled as belonging to target N. Table 5.2 shows the Log Erroneous Classification Metric for a typical frame in the indoor sequence.

The HMM smoothing stage may thus be useful if there are regions in the image which temporarily resemble one of the target objects, and so which may be discriminated against based on a model of pixel state transition characteristics, via a HMM. The HMM smoothing affects the pixel classification in such a way that the correctly classified pixels tend to be concentrated in the correct area. It may be useful in an algorithm such as the partitioned particle filtering with scan phase (described in the next chapter), where once the target becomes occluded, the scan phase causes the target model to converge rapidly on the best background approximation to the target. In the case where the Correct Classification Metric

	L*a*b*	L*a*b* with HMM
S_{T_0}	-356709.6	-985961.5
S_{T_1}	-302690.3	-826856.6
S_{T_2}	-474660.9	-1215590.0
S_{T_3}	-463249.5	-1204775.3
S_{T_4}	-731758.0	-1750928.8
S_{T_5}	-470384.0	-1225151.8
S_{T_6}	-400185.0	-1021655.5
S_{T_7}	-913469.5	-1035726.6

Table 5.2: Log Erroneous Classification Metric for each of the targets in the scene, at a typical frame. The addition of HMM smoothing has reduced the amount of incorrect classification.

is so similar to the actual observation method used per particle, however, we see that the effect on the pixel classification is detrimental with respect to the GMM based observation model currently being used.

Chapter 6

Importance sampling and Partitioned sampling

A well known extension to particle filtering is that of importance sampling. The idea is to concentrate the samples which represent a distribution at interesting regions within that distribution, without introducing bias by adjusting the particle concentration. Therefore, when we sample from an alternative proposal distribution, we immediately renormalize the proposed particle's weight using the knowledge of the probability of that particle's selection from our proposal distribution, then assign the observation density's weight to it, whereafter it represents a fair sample from the posterior distribution. With importance sampling comes higher computational efficiency, since we choose our proposal distribution in an intelligent way, possibly using the latest available data (as in [31]), from the current time step. In this regard, there is less reliance on the correctness of definition of the process dynamics, and one can even define dynamics within the importance distribution itself.

The idea of importance sampling is similar to the idea of importance reweighting, which is the basis of the partitioned particle filter.

6.1 Importance sampling

Suppose that within a recursive Bayesian particle filtering framework, we prefer not to sample from the prior distribution $P(\mathbf{X}_{t-1}|\mathbf{Z}_{t-1})$, but rather from an alternative proposal distribution $q(\mathbf{X}_t|\mathbf{Z}_t)$. We may find that in terms of the state expectation,

$$E[f(\mathbf{X}_t)] = \int f(\mathbf{X}_t)\frac{p(\mathbf{X}_t|\mathbf{Z}_t)}{q(\mathbf{X}_t|\mathbf{Z}_t)}q(\mathbf{X}_t|\mathbf{Z}_t)d\mathbf{X}_t$$
$$= \int f(\mathbf{X}_t)\frac{p(\mathbf{Z}_t|\mathbf{X}_t)p(\mathbf{X}_t)}{p(\mathbf{Z}_t)q(\mathbf{X}_t|\mathbf{Z}_t)}q(\mathbf{X}_t|\mathbf{Z}_t)d\mathbf{X}_t$$

$$= \int f(\mathbf{X}_t) \frac{w_t(\mathbf{X}_t)}{p(\mathbf{Z}_t)} q(\mathbf{X}_t|\mathbf{Z}_t) d\mathbf{X}_t$$

with

$$w_t(\mathbf{X}_t) = \frac{p(\mathbf{Z}_k|\mathbf{X}_t)p(\mathbf{X}_t)}{q(\mathbf{X}_t|\mathbf{Z}_t)},$$

where $w_t(\mathbf{X}_t)$ represents the weight of the sample drawn from the proposal distribution, after it has been renormalized to remove the bias introduced by the proposal distribution, and been observed against the data.

Then we have

$$E[f(\mathbf{X}_t)] = \frac{1}{p(\mathbf{Z}_t)} \int f(\mathbf{X}_t) w_t(\mathbf{X}_t) q(\mathbf{X}_t|\mathbf{Z}_t) d\mathbf{X}_t$$

$$= \frac{\int f(\mathbf{X}_t) w_k(\mathbf{X}_t) q(\mathbf{X}_t|\mathbf{Z}_t) d\mathbf{X}_t}{\int p(\mathbf{Z}_t|\mathbf{X}_t) p(\mathbf{X}_t) \frac{q(\mathbf{X}_t|\mathbf{Z}_t)}{q(\mathbf{X}_t|\mathbf{Z}_t)} d\mathbf{X}_t}$$

$$= \frac{\int f(\mathbf{X}_t) w_k(\mathbf{X}_t) q(\mathbf{X}_t|\mathbf{Z}_t) d\mathbf{X}_t}{\int w_t(\mathbf{X}_t) q(\mathbf{X}_t|\mathbf{Z}_t) d\mathbf{X}_t}$$

$$= \frac{E_{q(\mathbf{X}_t|\mathbf{Z}_t)}[w_t(\mathbf{X}_t) f(\mathbf{X}_t)]}{E_{q(\mathbf{X}_t|\mathbf{Z}_t)}[w_t(\mathbf{X}_t)]},$$

which shows us that we need to normalize at each time step across the new particle weights $w_t(\mathbf{X}_t)$ to form a posterior from which we may take the expectation.

So by drawing from $q(\mathbf{X}_t|\mathbf{Z}_t)$, we can approximate expectations of interest by the following:

$$E[f(\mathbf{X}_t)] = \frac{\frac{1}{N}\sum_{i=1}^{N} w_t(\mathbf{X}_t^{(i)}) f(\mathbf{X}_t^{(i)})}{\frac{1}{N}\sum_{i=1}^{N} w_t(\mathbf{X}_t^{(i)})} \approx \sum_{i=1}^{N} \widetilde{w}_t(\mathbf{X}_t^{(i)}) f(\mathbf{X}_t^{(i)}),$$

where the normalized weights are given by

$$\widetilde{w}_t(\mathbf{X}_t^{(i)}) = \frac{w_t(\mathbf{X}_t^{(i)})}{\sum_{j=1}^{N} w_t(\mathbf{X}_t^{(j)})}.$$

Using the state space assumptions of first order Markoveneity and observational independence given a state, the importance weights can be estimated recursively by

$$w_k = w_{k-1} \frac{p(\mathbf{Z}_t|\mathbf{X}_t)p(\mathbf{X}_t|\mathbf{X}_{t-1})}{q(\mathbf{X}_t|\mathbf{X}_{t-1}, \mathbf{Z}_t)}.$$

This may be slightly more obvious when we consider the original particle filter equation as presented in Chapter 3:

$$p(\mathbf{X}_t|Z_t) = k_t p(\mathbf{Z}_t|\mathbf{X}_t) \int_{\mathbf{X}_{t-1}} p(\mathbf{X}_t|\mathbf{X}_{t-1}) p(\mathbf{X}_{t-1}|Z_{t-1}) d\mathbf{X}_{t-1}.$$

When we introduce the importance dynamics, this becomes:

$$p(\mathbf{X}_t|Z_t) = k_t p(\mathbf{Z}_t|\mathbf{X}_t) \int_{\mathbf{X}_{t-1}} \frac{p(\mathbf{X}_t|\mathbf{X}_{t-1})}{q(\mathbf{X}_t|\mathbf{X}_{t-1})} q(\mathbf{X}_t|\mathbf{X}_{t-1}) p(\mathbf{X}_{t-1}|Z_{t-1}) d\mathbf{X}_{t-1}.$$

In [7], the following algorithm for the implementation for importance sampling is presented (although it is not the only way to implement importance sampling), and allows for the use of multiple proposal densities. The selection of each proposal density itself may be randomly selected. The term "initialization prior" refers to the initial distribution on the state as determined by an importance distribution, and would be specified manually, for example by a Gaussian around the expected expectation value. The importance correction factor is there to remove the bias introduced by sampling from an importance distribution. Assume that we may sample from the prior, the importance initialization prior, or the importance dynamics. Then, for each n^{th} of N samples:

- First generate a random number $\alpha \in [0, 1)$, from the uniform distribution in that interval.
- Sample from the prediction density $p(\mathbf{X}_t|Z_{t-1})$ as follows:
  1. If $\alpha < q$ use the initialization prior. Choose $\mathbf{s}_t^{(n)}$ by sampling from $g_t(\mathbf{X}_t)$ and set the importance correction factor $\lambda_t^{(n)} = 1$.
  2. If $q \leq \alpha < q + r$ use importance sampling. Choose $\mathbf{s}_t^{(n)}$ by sampling from $g_t(\mathbf{X}_t)$ and set $\lambda_t^{(n)} = f_t(\mathbf{s}_t^{(n)})/g_t(\mathbf{s}_t^{(n)})$, where
  $$f_t(\mathbf{s}_t^{(n)}) = \sum_{j=1}^{N} \pi_{t-1}^{(j)} p(\mathbf{X}_t = \mathbf{s}_t^{(n)}|\mathbf{X}_{t-1} = \mathbf{s}_{t-1}^{(j)}).$$
  3. If $\alpha \geq q + r$ generate a sample from the prior $p(\mathbf{X}_{t-1}|\mathbf{Z}_{t-1})$ and then sample from $p(\mathbf{X}_t|\mathbf{X}_{t-1} = \mathbf{s}_{t-1}^{(n)})$ and set $\lambda_t^{(n)} = 1$.
- Calculate the observation density, and use this to weight the new sample according to the observed image data $\mathbf{Z}_t$ and the importance sampling correction term.
  $$\pi_t^{(n)} = \lambda_t^{(n)} p(\mathbf{Z}_t|\mathbf{X}_t = \mathbf{s}_t^{(n)})$$
- After this, normalize the sample weights such that $\sum_n \pi_t^{(n)} = 1$.

It can be said that the basic particle filter is a special case of the particle filter with importance sampling, where the importance distribution is equal to the post-dynamics distribution of the state variable, so that $q(\mathbf{X}_t|\mathbf{Z}_t) = p(\mathbf{X}_t|\mathbf{X}_{t-1})$.

The design of the proposal distribution is crucial to the efficient functioning of a particle filter. In addition there are two constraints which must be imposed on the selection of this proposal distribution. Firstly that the distribution must have the same region of support as the prior, and secondly, the distribution should be affected by the most recent observations.

The problem with any formulation of Sequential Importance Sampling (SIS) is that the sample variance typically increases over time until the effective number of particles (N_{eff}) decreases to one. For this reason at every few time steps, or perhaps every time step, we perform a resampling step.

6.1.1 Resampling

In the resampling operation, N unequally weighted particles are resampled into a new set of N equally weighted particles:

$$\{\mathbf{s}_t^{(i)}, \pi_t^{(i)}\}_{i=1}^{N} \longrightarrow \{\mathbf{s}'^{(j)}_t, \frac{1}{N}\}_{j=1}^{N}$$

Here the i_{th} particle $\mathbf{s}_t^{(i)}$ is chosen with probability $\pi_t^{(i)}$, and then stored in the new distribution as $\mathbf{s}'^{(j)}_t$ with probability $\frac{1}{N}$, with $j = 1..N$.

The result of the algorithm is that from the previous sample set $\{\mathbf{s}_t^{(n)}, \pi_t^{(n)}\}, n = 1, ..., N$ at time step t, a new sample set $\{\mathbf{s}'^{(n)}_t, \frac{1}{N}\}, n = 1, ..., N$ is constructed, and this procedure may be repeated whenever the number of effective particles decreases below a threshold. The number of effective particles is a measure of how many particles are used per time step, on the basis that those which are not used are wasted. The resampling operation may be performed to keep this number high. For a more detailed explanation on the number of effective particles, see the next chapter.

6.2 Partitioned Sampling

Partitioned sampling for particle filtering was introduced in [33] and in [32]. The mechanism of partitioned sampling can best be expressed in using a block diagram formulation. It consists of stages of importance sampling followed by a final observation stage for each time step. One step of conventional condensation can be conveniently expressed with the following diagram:

$\boxed{p(\mathbf{X}_{t-1}|Z_{t-1})} \rightarrow \boxed{\sim} \rightarrow \boxed{*h(\mathbf{X}'|\mathbf{X})} \rightarrow \boxed{\times f(\mathbf{Z}_t|\mathbf{X}')} \rightarrow \boxed{p(\mathbf{X}|\mathbf{Z}_t)}$

In this diagram, $\sim$ indicates a resampling step. The $*$ operation indicates that the resampled distribution undergoes adjustment by the convolution dynamics (function $h(\mathbf{X}'|\mathbf{X})$) and finally the samples undergo a measurement stage, denoted by $\times f(\mathbf{Z}_t|\mathbf{X}')$.

6.2.1 Weighted resampling

The weighted resampling operation is a method for refining the concentration of samples which represent a probability distribution at specific locations within that distribution, without introducing bias to those regions, i.e. without changing the distribution which the samples represent.

Let $g(\mathbf{X})$ be a strictly positive and continuous function over the available sample space $\mathbf{X}$. Given a particle set $\{\mathbf{s}_t^i, \pi_t^i\}$, with $i = 1..N$, a weighted resampling operation will produce a new set $\{\mathbf{s}'^i_t, \pi'^i_t\}$, with $i = 1..N$. The method according to which this new sample set is generated is as follows: First calculate importance weights $\rho_i = g(\mathbf{s}_t^i)/\sum_{j=1}^n g(\mathbf{s}_t^j)$. Then select the indices of the particles from this set, $k_1, k_2, .., k_n$ by setting $k_i = j$ with probability ρ_j. Finally, set $\mathbf{s}'^i_t = \mathbf{s}_t^{k_i}$ and $\pi'^i_t = \pi_t^{k_i}/\rho_{k_i}$. Weighting the samples in this way removes any bias introduced when sampling them according to the artificially applied probability weights ρ_i.

A simple example of partitioned sampling with two partitions may be as follows:

$$\boxed{p(\mathbf{X}|Z_{t-1})} \rightarrow \boxed{\sim} \rightarrow \boxed{*h(\mathbf{X}'|\mathbf{X})} \rightarrow \boxed{\sim g_1}$$
$$\rightarrow \boxed{*h(\mathbf{X}''|\mathbf{X}')} \rightarrow \boxed{\times f(\mathbf{Z}_t|\mathbf{X}'')} \rightarrow \boxed{p(\mathbf{X}|Z_t)}$$

Where in this diagram, $\sim g$ represents a weighted resampling step using the function g. We see also that each partition has an associated convolution dynamics operation in which a subspace of any sample may be affected, depending on which partition the sample is currently in. There is also given a more complicated example of partitioned sampling with multiple partitions:

$$\boxed{p(\mathbf{X}|Z_{t-1})} \rightarrow \boxed{\sim} \rightarrow \boxed{*h(\mathbf{X}'|\mathbf{X})} \rightarrow \boxed{\sim g_1}$$
$$\rightarrow \boxed{*h(\mathbf{X}''|\mathbf{X}')} \rightarrow \boxed{\sim g_2}$$
$$\rightarrow \boxed{*h(\mathbf{X}'''|\mathbf{X}'')} \rightarrow \boxed{\sim g_3}$$
$$\rightarrow \boxed{*h(\mathbf{X}''''|\mathbf{X}''')} \rightarrow \boxed{\sim g_4}$$
$$\rightarrow \boxed{*h(\mathbf{X}'''''|\mathbf{X}'''')} \rightarrow \boxed{\times f(\mathbf{Z}_t|\mathbf{X}'''')} \rightarrow \boxed{p(\mathbf{X}|Z_t)}$$

In this example we see that we can introduce as many partitions as we would like. Typically the number of partitions depends on the degree to which the parameters or groups of parameters which make up the parameter vector of each particle can be evaluated independently. Also useful are the diagrams in Fig. 6.1, Fig. 6.2, Fig. 6.3 and Fig. 6.4 which depict the iterative refinement of the region of interest in the state space, which results from the partitioning.

6.2.2 Partitioned sampling for articulated objects

In [33], where the partitioned sampling paradigm is applied to the problem of tracking an articulated object, this approach works because the shape of the hand can be partitioned into the base of the hand (fist), the thumb, and each finger, totalling six partitions for a

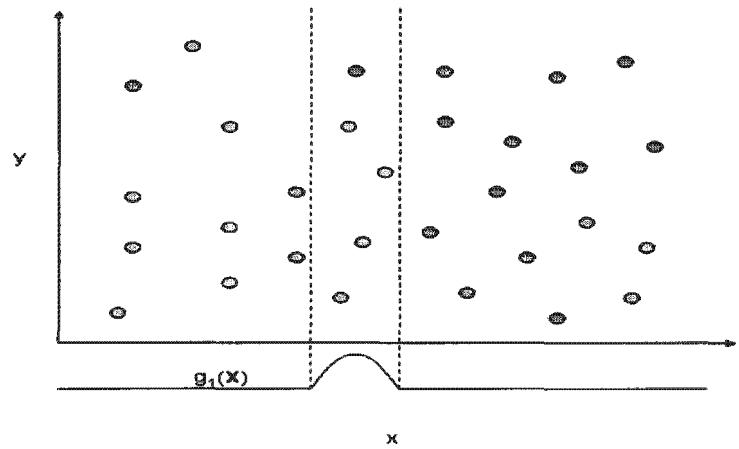

Figure 6.1: A 2D state space is resampled in its first dimension according to a weighting function $g_1(\mathbf{X})$.

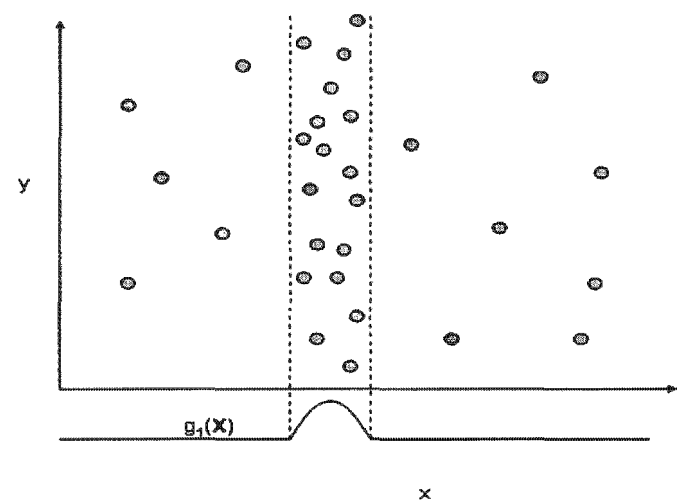

Figure 6.2: The weighting function multiplies the number of particles falling in the region where the weighting function is maximal. The bias introduced at this stage is removed by reassigning the weights for the new particles as described elsewhere. Since the number of samples stored per partition per time step is usually constant, this means we lose some of the other samples.

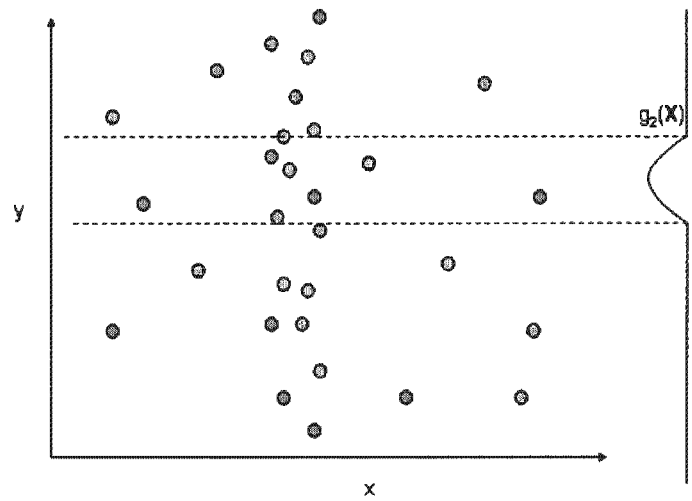

Figure 6.3: On this new sample set is applied the second weighting function, $g_2(\mathbf{X})$.

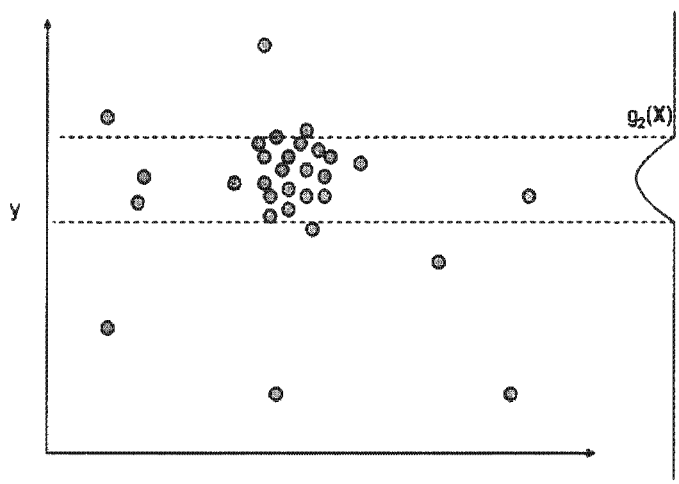

Figure 6.4: Once again the particles selected by the weighting function are multiplied.

single hand. In this way, samples in the probability space representing the components of the hypothesized hand are first concentrated, without introducing bias, in areas where the base of the hand is likely to be. In the next partition, the probability distribution is further refined, this time in such a way as to increase the number of samples in the region of space where samples are likely to successfully represent the location of the thumb, and so on for each finger. When we arrive at the last partition, we will ideally have a set of samples which represent the same distribution as we had in the first partition, (allowing for the convolution dynamics), but with a higher concentration of samples in the region of space where we expect the correct configuration to apply.

These samples are then measured against the true observation density $P(\mathbf{Z}_t|\mathbf{X}_t)$ and with these new weights, will form the sample set representing the first partition of the next time step. Also useful in the understanding of partitioned sampling are Fig. 6.5 which shows the necessary operations, and Fig. 6.6 which shows at which stages sample sets need to be stored in this implementation of partitioned particle filtering.

6.2.3 Partitioned sampling for person tracking

In their earlier work MacCormick et al. [32], used partitioned sampling to track the contours which the heads of people made against the background, using a contour based observation, and tracking in image coordinates. In this implementation, the tracking of entire people is done in world coordinates using colour modelling information.

Using an assumption that the components of the parameter vector represented by each sample may be independently observed for likelihood, we partition the sample space in such a way that we have one partition per person per frame. This assumption is not exactly correct as the components of the parameter vector of each sample may indeed affect the way in which others parts of the vector should be measured or observed, i.e. it affects the weighting function that should be used to measure the sample at that point. However, it does allow for robust tracking of people in a video sequence.

6.2.4 Varying numbers of partitions

In fact the number of people per frame in a video sequence may vary in time as people enter and leave the scene, so we may expect the number of partitions to vary dynamically with the number of people. This is possible if there is a way of determining the number of people in the scene at any given point in time. In this implementation the algorithm is updated by manually setting the number of targets to track in the current frame, and the

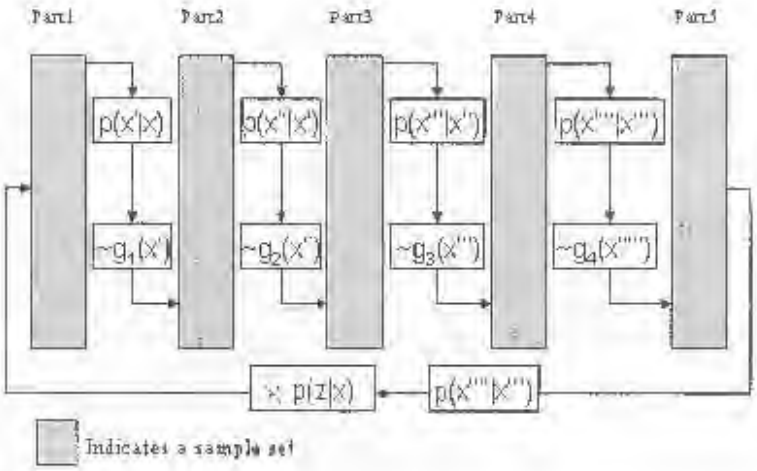

Figure 6.5: This is a block diagram of the partitioned sampling process, with the stages at which we need to store samples shown.

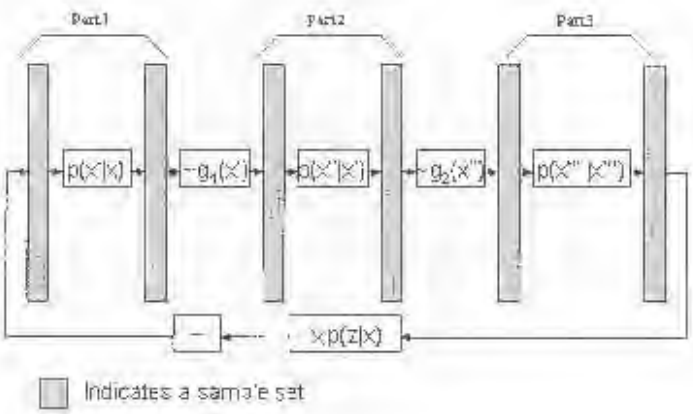

Figure 6.6: We can divide the processing of each partition into two stages: the dynamics, and the weighted resampling.

number of partitions varies accordingly.

6.2.5 Weighting functions

Our next goal is to define a reasonable set of weighting functions based on the observation values of particular components of the particle set. Each weighting function should evaluate a particle from this set according to these components, and not any other components, or the usefulness of partitioning the search space in a particular way is nullified.

The observation of particular components of a particle should be affected by other components insofar as those components affect the visibility of the component in question via occlusion or the mode of observation or otherwise, however those components should not be evaluated themselves, and this is the crucial difference.

In our implementation, the weighting function seeks to determine with respect to the components of the particle in question, how likely it is that this component of this particle gave rise to the observed data.

We have already stated that our particle vector consists of the concatenated x, y and z world coordinates of all target objects, and that the mode of observation for such a hypothesis is to render all ellipsoids and calculate the probability of the resultant pixel labelling against the image data.

In this implementation, the weighting function used per person is similar to the observation method for the entire particle:

$$g_{l_i}(\mathbf{X}_t) = g_{l_i}(\mathbf{Z}_t|\mathbf{X}_t) = \prod_{g=1}^{G} g_{l_i}(z_g|\mathbf{X}_t) = \prod_{g\in R} p_{l_i}(z_g) \times \prod_{g\in\overline{R}} (1 - p_{l_i}(z_g))$$

where R indicates the set of pixels which have been labelled as belonging to the object which corresponds to l_i. The rest of the pixels $\overline{R}$ are those labelled as belonging to all other target objects (including the background). In this way we may measure the correctness of a particular component of a particle, which hypothesizes that a particular object is in a particular place, with a particular amount of occlusion, and which hypothesizes that all other parts of the image do not belong to that object, but does not specify to which objects they do belong. Note that occlusion has been calculated for the target in question already, at the ellipsoid rendering stage. If the target is entirely occluded, then the region R will be empty. A similar normalization may be done as before to allow us to visit only the region in the image occupied by the target of the current partition, so that:

$$g_{l_i}(\mathbf{X}_t) \propto \frac{\prod_{g\in R} p_{l_i}(z_g)}{\prod_{g\in R}(1 - p_{l_i}(z_g))}$$

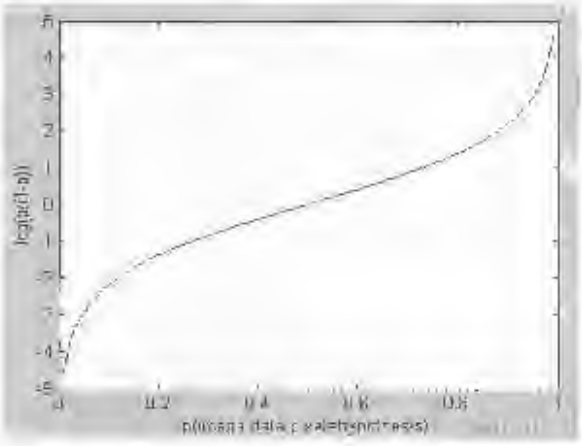

Figure 6.7: The contribution of a single pixel to the weighting function depending on its probability of classification as a pixel from a particular model.

As before, we may convert this multiplication to a summation in the log domain:

$$\begin{aligned}\log(g_{l_2}(\mathbf{X}_t)) &\propto \log\left(\prod_{q\in R}\frac{p_{l_q}(z_q)}{(1-p_{l_q}(z_q))}\right)\\ &= \sum_{q\in R}\log\left(\frac{p_{l_q}(z_q)}{1-p_{l_q}(z_q)}\right).\end{aligned}$$

Then, the contribution of a single pixel to this sum, with respect its probability of having been correctly classified, may be seen in Fig. 6.7.

6.3 Noise and Dynamics for each partition

As mentioned before, this implementation does not model the state space dynamics in any particularly complex way. Simple diffusion dynamics are used, with random Gaussian noise being added to the sample set at each partition before the importance reweighting is applied. We therefore introduce the diffusion dynamics for the components of the state vector immediately after the resampling operation for each new partition, as suggested in the original formulation in [32] and [33].

There is nothing to prevent the use of a more sophisticated dynamical model at this stage.

6.4 Scan phase

This implementation of the partitioned particle filter also contains a novel method for recovering from tracking errors. In the problem of tracking people, we have additional constraints on the search space, namely that any person may occupy one of a set of known points, typically along the surface of the floor on which that person is walking. Although it would not be feasible to search exhaustively through all possible combinations of locations for all people, it is feasible and useful at each partition to allocate a small fixed number of samples drawn randomly from the prior distribution, but altered to reflect the possibility that the target object to which the partition has been allocated may occupy a different space on the floor of the room (which corresponds to particular areas in the state space).

Points at fixed locations on the floor are represented by this small set of samples, with the goal of tracking the person should he walk through any of these sensitive zones, these locations in the state subspace which we insist on observing.

The bias introduced by searching with this alternative, deterministic method can be removed, since we can evaluate the probability of drawing this sample from the prior, via the dynamics. Its position in state space is a combination of some draw from the distribution represented by the samples at the previous partition, and values from a database of target coordinates which overwrite certain of the elements of that state vector before its observation, in a similar way to the crossover operator presented in [32], and the genetic modifiers presented in [28].

The partitioned sampling algorithm augmented with this scan phase performs somewhat better than without it, as it allows for the recovery of lost targets. In fact, this scan phase of the algorithm allows the tracker, if supplied with the correct tracking and background models, to locate all the targets and stabilize around the correct joint hypothesis within a few number of frames (given that the target objects all become visible at some stage) even when the tracker is randomly initialized.

6.4.1 Deterministic sequences within the Bayesian paradigm

We now briefly present a description of the way in which a set of samples which are to be specifically included, and processed deterministically (i.e. all the samples in this distribution will be processed) at a particular partition in a particle filter, may be combined in a weighted sum with the prior (after partition level dynamics) distribution, and thus processed as a part of an importance mixture distribution.

Consider a prior which has undergone dynamics, $p(\mathbf{X}'|\mathbf{X})p(\mathbf{X}|Z)$, and an importance

distribution $g(\mathbf{X}') = \sum_i \delta(\mathbf{X}' - \mathbf{x}_i)$, consisting of a collection of Dirac deltas. We take a weighted sum of these two distributions, forming a new importance distribution, $G(\mathbf{X}') = ap(\mathbf{X}'|\mathbf{X})p(\mathbf{X}|Z) + bg(\mathbf{X}')$, where $a + b = 1$. To remove the bias introduced when using this importance distribution as a proposal distribution, we then normalize each sample in the combined pdf with $f_t(\mathbf{s}_t^{(n)})/G(\mathbf{X}')$, where

$$f_t(\mathbf{s}_t^{(n)}) = \sum_{j=1}^{N} \pi_{t-1}^{(j)} p(\mathbf{X}_t = \mathbf{s}_t^{(n)} | \mathbf{X}_{t-1} = \mathbf{s}_{t-1}^{(j)}).$$

When calculating $1/G(\mathbf{X}')$ for the samples taken from the prior, we find that

$$\frac{1}{G(\mathbf{X}')} = \frac{1}{p(\mathbf{X}'|\mathbf{X})p(\mathbf{X}|Z) + g(\mathbf{X}')} \approx \frac{1}{p(\mathbf{X}'|\mathbf{X})p(\mathbf{X}|Z)},$$

since the samples taken from the prior are almost certainly not also in the importance distribution $g(\mathbf{X}')$. For the samples of the importance distribution $g(\mathbf{X}')$, we find that

$$\frac{1}{G(\mathbf{X}')} = \frac{1}{p(\mathbf{X}'|\mathbf{X})p(\mathbf{X}|Z) + g(\mathbf{X}')}$$

always, since the importance distribution is by design requirement on a region which has the support of the prior.

We may now describe the scan phase augmentation in the following algorithm:

Assume that there are B basic particles and A augmented particles per partition. In addition, assume that to the distribution represented by the B basic particles, we assign the weight b, and to the distribution of auxiliary particles the weight a, as per the discussion in the previous subsection. Then for each each partition which is not the final partition:

- First subsample via the dynamics $h(\mathbf{X}'|\mathbf{X})$ to obtain from the pre-dynamics set $\{\mathbf{s}_t^i, \pi_t^i\}$ with $i = 1..(B+A)$ a smaller, post dynamics set $\{\mathbf{s}'^i_t, \pi_t^i\}$, with $i = 1..B$.
- Sample from this post dynamics set to attain another set of auxiliary particles $\{\mathbf{s}^{\mathbf{a}i}_t, \pi_t^i\}$, with $i = 1..A$.
- Adjust each of the particles in this new set $\{\mathbf{s}^{\mathbf{a}i}_t, \pi_t^i\}$, with $i = 1..A$, in a particular way to contain in the elements of each sample corresponding to the current partition a new set of coordinates. So $\{\mathbf{s}^{\mathbf{a}i}_t, \pi_t^i\}$, with $i = 1..A$, becomes $\{\mathbf{s}^{\mathbf{w}i}_t, \pi_t^i\}$, with $i = 1..A$. The way we should weight these new particles is important for the maintenance of the recursive Bayesian paradigm. In particular, we weight each sample in the smaller, post dynamics set $\{\{\mathbf{s}'^i_t, \pi_t^i\}$, with $i = 1..B\}$ with $\lambda^{(i)} = f_t(\mathbf{s}_t^{(n)})/(b * p(\mathbf{X}'|\mathbf{X}))$. The auxiliary particles are weighted with $\lambda^{(i)} = f_t(\mathbf{s}_t^{(a)})/(b * p(\mathbf{X}'|\mathbf{X}) + a * A)$ where b is the relative weighting given to the importance distribution, and a the weighting given to the dynamics on the prior, in the mixture distribution.
- Calculate the importance reweighting function $g_x(\mathbf{X})$ on each of the samples in the combined set of basic particles and augmented particles, in the current partition, remembering that the previous weight of any particle is also factor in its new weight (this is where the factor b may come into play). So the new weight of any particle i is $g_x(\mathbf{X}) * \lambda^{(i)}$, where $g_x(\mathbf{X})$ is the importance reweighting function for the current partition.
- A weighted resampling into the new sample set at the next partition is then performed.

If we are in the final partition, the samples in this set are measured against the observation density $p(\mathbf{Z}_t|\mathbf{X}_t)$, then resampled into the first partition of the next time step.

The scan phase of the algorithm incorporated into the partitioned particle filter is shown in Fig. 6.8, Fig. 6.9, and Fig. 6.10. In Fig. 6.8 we see at which stages of the algorithm the augmented particles are inserted. In Fig. 6.9 we see how at each partition, the sample set is

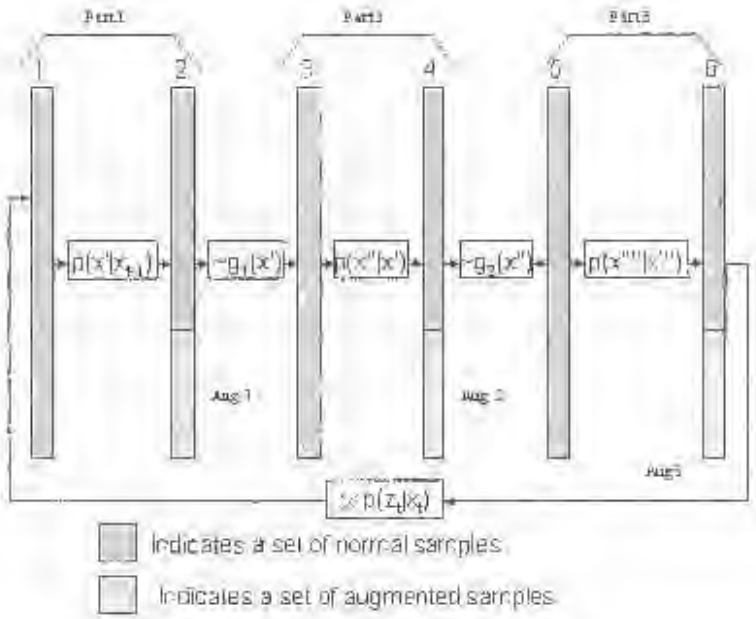

Figure 6.8: We add to certain of the sample stages an auxiliary set of samples taken from a distribution which is partially based on the posterior of the previous partition and partially determined by hand.

first subsampled via the dynamics into a smaller sample set, and then the remainder of the sample set is filled with augmented particles, shown in Fig. 6.10, using a database of state space coordinates which are used in the creation of the augmented particles. In Fig. 6.11, Fig. 6.12 and Fig. 6.13, we see the points on the surface of the floor to which components of the augmented particles are assigned, in their state spaces.

6.5 Reassigning the partitions

6.5.1 Expanding and Contracting

A simple but relevant issue that a partitioned particle filter implementation has to deal with when assigning partitions to target objects, is that the target objects are likely to arrive and leave on the scene in quite a random order. The partitions thus need to be allocated and reallocated to the respective target objects on the fly, in such a way that the number

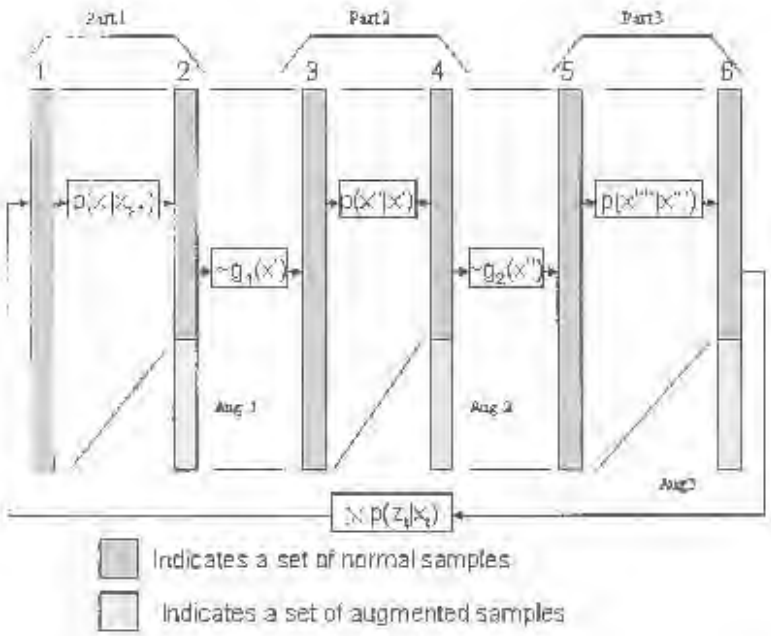

Figure 6.9: At each partition, the entire set, namely the base set and auxiliary set of particles are all evaluated by the weighted resampling function and passed into the next partition. During the dynamical stage however, the prior at each partition is sub-sampled into the posterior distribution, to make space for the auxiliary particle set, which forms a small part of the partition posterior, on which the next weighting function must be performed.

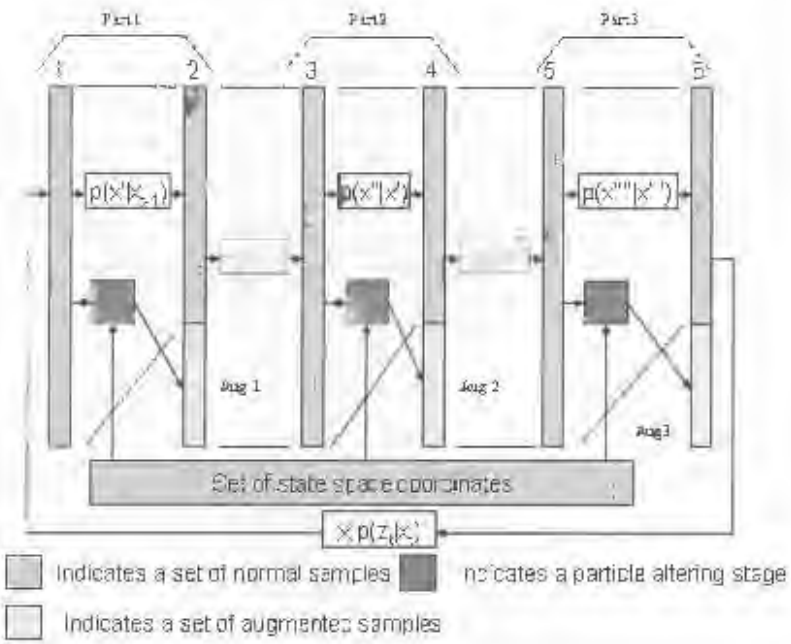

Figure 6.10: This diagram illustrates the way in which samples are modified before they are stored in the auxiliary particle set in the posterior of each partition. Particles in the auxiliary particle set are sampled from the dynamics of the partition prior, but have some of their components altered and set to specific state space coordinates.

Figure 6.11: After the basic particles have been processed, the augmented particles are processed for the scan phase. The location for a particular target represented in the particles in the scan phase is at a predetermined points on the floor of the intelligent environment.

Figure 6.12: After the basic particles have been processed, the augmented particles are processed for the scan phase. The location for a particular target represented in the particles in the scan phase is at a predetermined points on the floor of the intelligent environment.

Figure 6.13: After the basic particles have been processed, the augmented particles are processed for the scan phase. The location for a particular target represented in the particles in the scan phase is at a predetermined points on the floor of the intelligent environment.

of partitions used expands and contracts with the number of target objects.

6.5.2 Best targets first

Another issue which would improve the issue of such a tracker is to assign to the first partition an object which can with the greatest certainty be tracked correctly, as opposed to some target object which has been temporarily occluded. The reasoning for this is as follows. Suppose the first partition were assigned to an occluded target object. Then the weighted resampling would return a set of samples to the second partition which could not possibly correspond to the correct position of the first object. In the next partition, these hypotheses are carried through and then cause erroneous occlusion within the joint hypothesis as caused by the badly-tracked first target object.

The lack of complete observational independence with respect to the components of the state vector representing each object has a negative impact on the tracking of all objects if we begin the search space partitioning using a weighting function which will partition the search space badly due to the corresponding object's occlusion.

One way to improve the tracker based on this observation is to allocate partitions to objects in such a way that the first partition is allocated to the object which is most likely to be visible, the second partition to the second most likely target, and so on. Of course, this implies the need for some kind of occlusion reasoning framework according to which

the partitions may be allocated. A simple solution for this is to sort the objects in terms of the depth (distance from camera center) via the expectation values for each particle's location in the state space of the posterior state distribution for the previous time step.

In fact, this best-first approach was not implemented here, and the results therefore suffer from the aforementioned problems. The tracking is, however, relatively stable even without this potential improvement.

6.6 Flow Chart of algorithm

We are now in a position to to describe the algorithm as a whole in terms of a flow chart, as in Fig. 6.14. As indicated by the chart, the HMM smoothing phase causes the particle filtering stage to run 7 frames behind the current frame. If this is too much, the HMM smoothing can be done using fewer than 15 frames, or if the pixel state dynamics are such that the pixel classification will not benefit from HMM smoothing, it can be removed completely. The block containing the particle filtering posterior distribution update may be implemented using the basic particle filter, the partitioned particle filter, or the partitioned particle filter with scan phase.

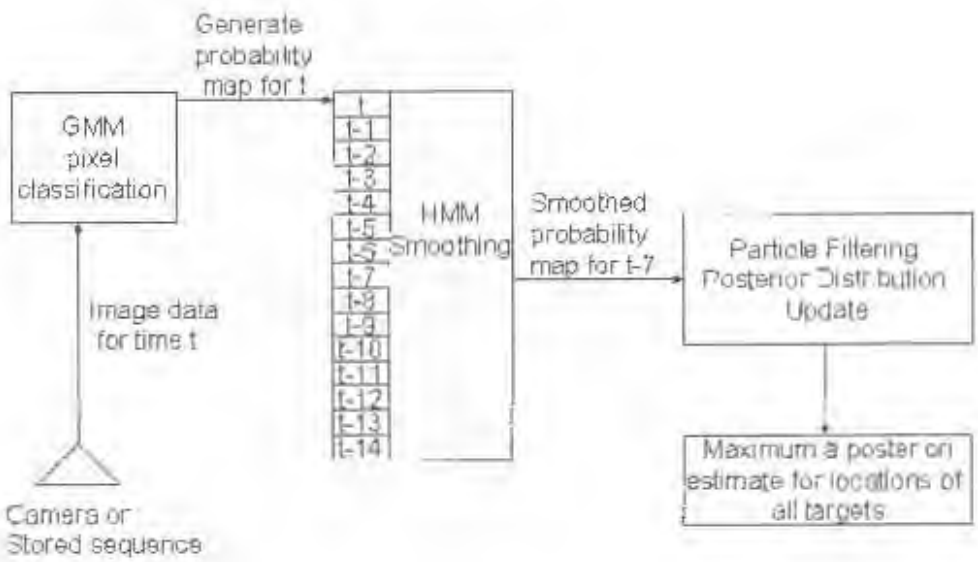

Figure 6.14: Flow chart for algorithm

Chapter 7

Results

In this chapter we present some graphs and performance statistics of the functioning and performance of partitioned sampling with GMMs pixel level classification and HMM smoothing, to track people interacting in complex ways in a static scene, as well as some results of people interacting naturally in a natural scene.

Within the literature about particle filters for visual tracking, and indeed in the literature for visual tracking algorithms in general, there is no standard way of measuring the tracking accuracy of a particle filter implementation, or indeed of measuring the accuracy of any visual tracking algorithm.

Moreover, due to the resampling stage which occurs once per partition in the algorithm developed in this implementation, the usual metrics of the accuracy of the selection of the regions of state space for sampling such as the Effective Number of Particles N_{eff} and the Survival Rate ([33]) are not useful except at the level of individual partitions. We may however examine the relationship between this measure and between a ground-truth statistic such as the one developed later in this chapter.

Also provided are graphs illustrating the tracking statistics using the basic particle filter, which we shall see performs very poorly, and the partitioned particle filtering algorithm without the scan phase augmentation.

7.1 Partitioned Particle Filter performance assessment based on ground-truth data

We can plot the correct percentage of pixel labelling produced by the tracker at any stage. This is possible by using manually labelled data from the same scene, and we propose that unless the world coordinates of the target object are known to high accuracy, this is a valid and appropriate empirical measure for the functioning of a tracking algorithm.

For every frame, and for every object, we may generate two statistics. The first is the percentage of manually labelled pixels which are correctly labelled by the Maximum a Posteriori (MAP) likelihood of the state estimate (per frame). The second is the percentage of the pixels labelled as belonging to a certain object by the MAP estimate which in fact do belong to the object. These statistics should be used together, since using either one of them alone may lead to a false appraisal of a particular MAP estimate. For example in the case where the hypothesized location of an object is too close to the camera center, it will generate a particularly large area in the corresponding image space. The large area of this hypothesis may then cause the reference segmentation to be fully covered, although the hypothesis is potently bad. This will be revealed by the second statistic. Conversely, the second statistic would on its own indicate a good match given a MAP estimate which was too far away from the camera center, with a corresponding hypothesis region in image space which was too small. All the pixels of the MAP estimate may then correctly match up to the manually segmented data, and only the first statistic would actually reveal the inaccuracy of the MAP estimate for the target object in question.

7.1.1 Performance of partitioned particle filter with scan phase

The graphs in Fig. 7.1, Fig. 7.4, Fig. 7.7, Fig. 7.10, Fig. 7.2, Fig. 7.5, Fig. 7.8, Fig. 7.11, Fig. 7.3, Fig. 7.6, Fig. 7.9 and Fig. 7.12 describe the tracking success of the partitioned particle filter, using two hundred basic samples and a hundred augmented samples, for the tracking scene in which there are seven highly colourized people. The characteristics for the tracking of the first two people (white and light-blue) are shown in Fig. 7.1, Fig. 7.2 and Fig. 7.3. The characteristics for the tracking of the second two people (orange and green) are shown in Fig. 7.4, Fig. 7.5 and Fig. 7.6. The characteristics for the tracking of the third two (red, dark-blue) are shown in Fig. 7.7, Fig. 7.8 and Fig. 7.9. The characteristics for the tracking of the last target (Yellow) are shown in Fig. 7.10, Fig. 7.11 and Fig. 7.12.

The best way to interpret these graphs is to look at the amount of time each tracking curve is above zero, in comparison to the number of times each curve falls to the bottom (zero percent successful pixel classification per person). So long as a curve in the graph is above zero, it indicates that the target object is still being correctly, although possibly inaccurately, tracked. If the percentage successful pixel classification falls to zero, the target object is essentially lost. It is either random noise, the target walking into the tracker's scope, or the scan phase of the algorithm which causes the successful reacquisition of each target. Notice that all targets are lost at some stage, but always reacquired. In fact, for each of the target objects which are tracked from their arrival on the scene, the tracker is still locked onto their locations when they leave the scene. Another interesting point about

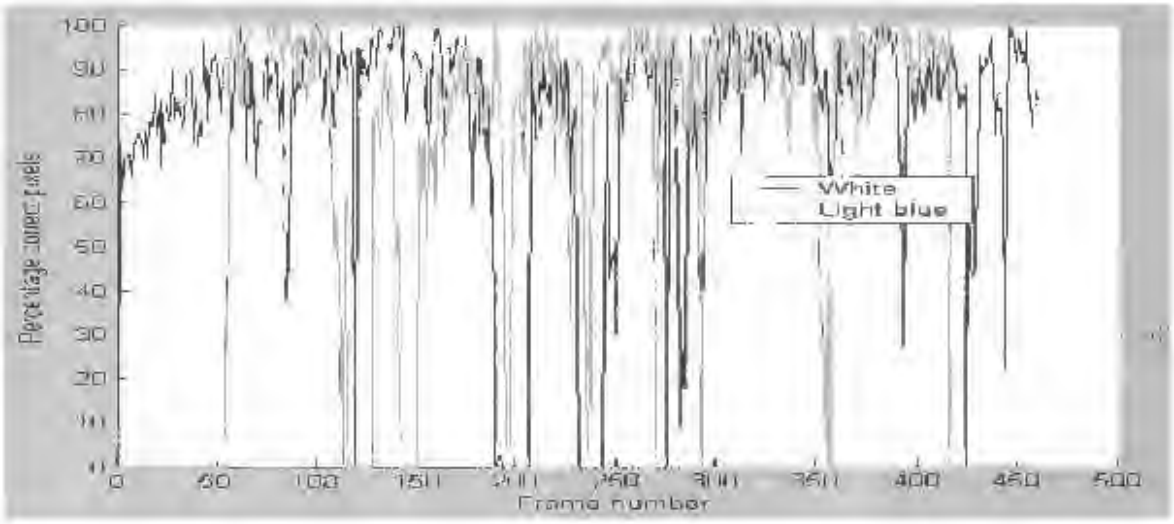

Figure 7.1: Graph of tracking of white and light-blue targets using the percentage of manually labelled pixels correctly placed in the hypothesis region for each person.

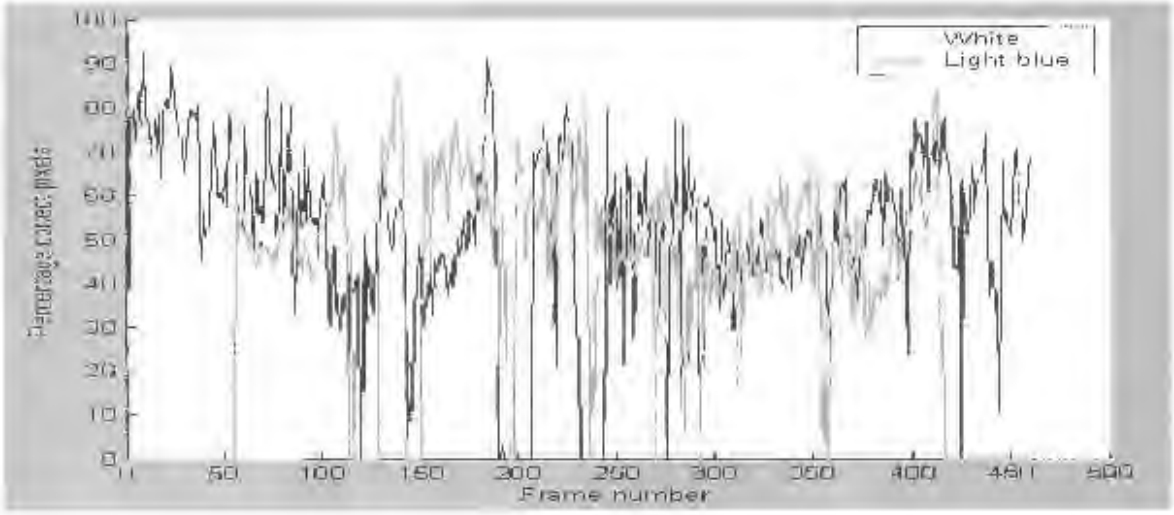

Figure 7.2: Graph of tracking of white and light-blue targets using the percentage of hypothesized pixels which are correctly labelled as per the manually segmented data.

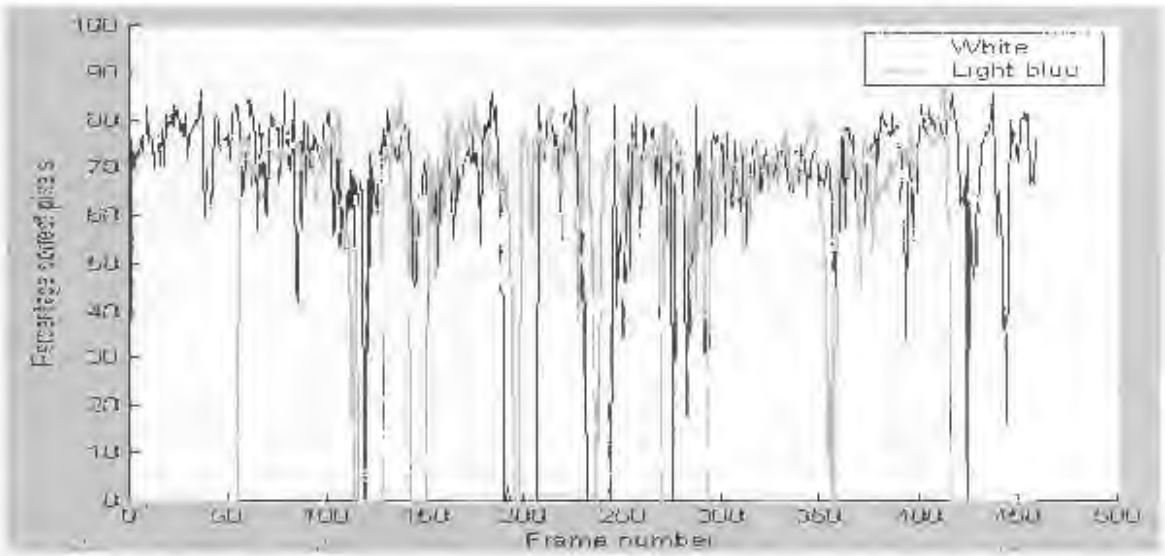

Figure 7.3: The average of the percentages shown in Fig. 7.1, and Fig. 7.2.

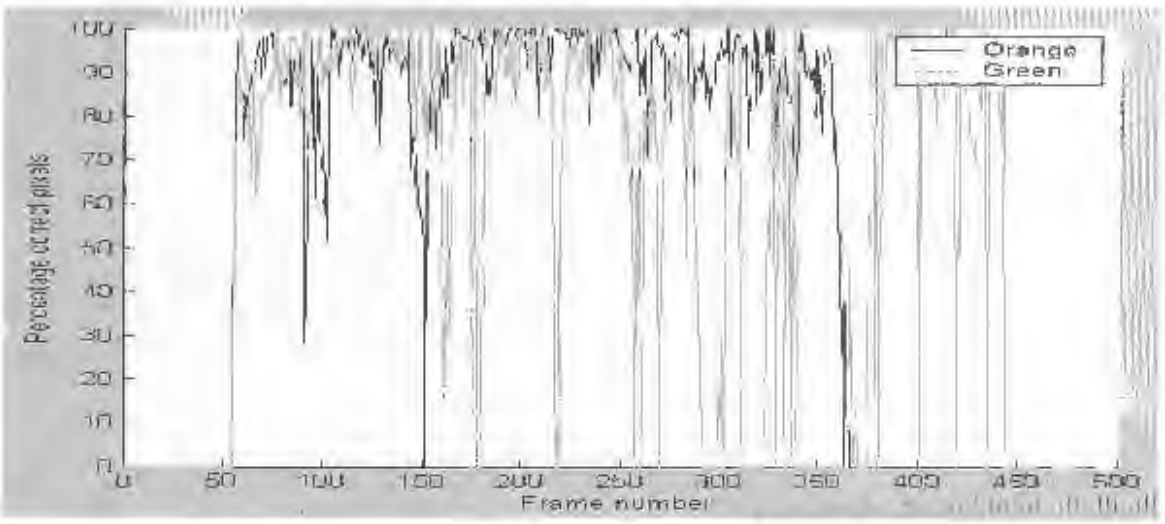

Figure 7.4: Graph of tracking of orange and green targets using the percentage of manually labelled pixels correctly placed in the hypothesis region for each person.

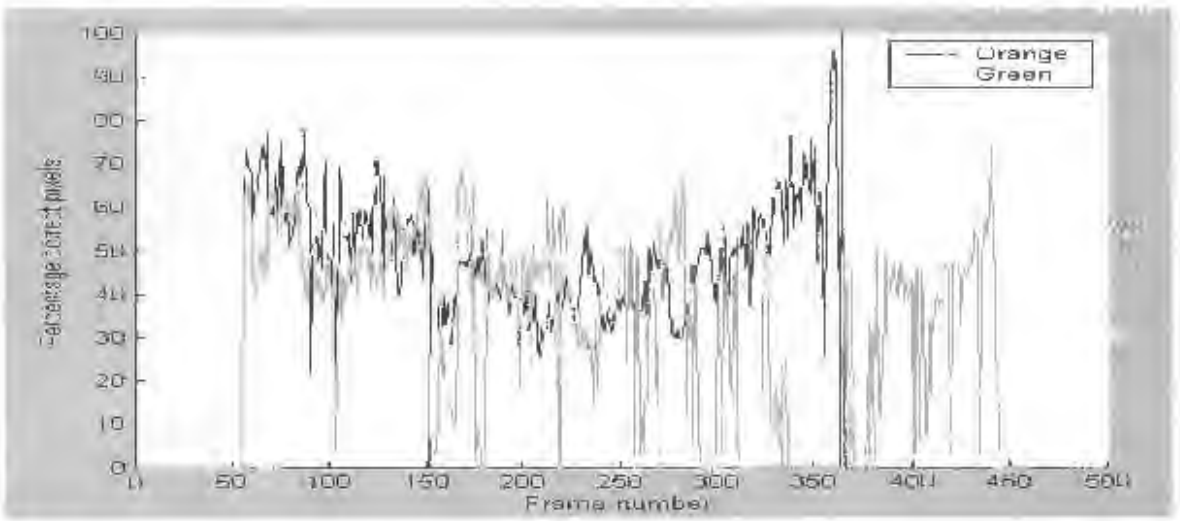

Figure 7.5: Graph of tracking of orange and green targets using the percentage of hypothesized pixels which are correctly labelled as per the manually segmented data.

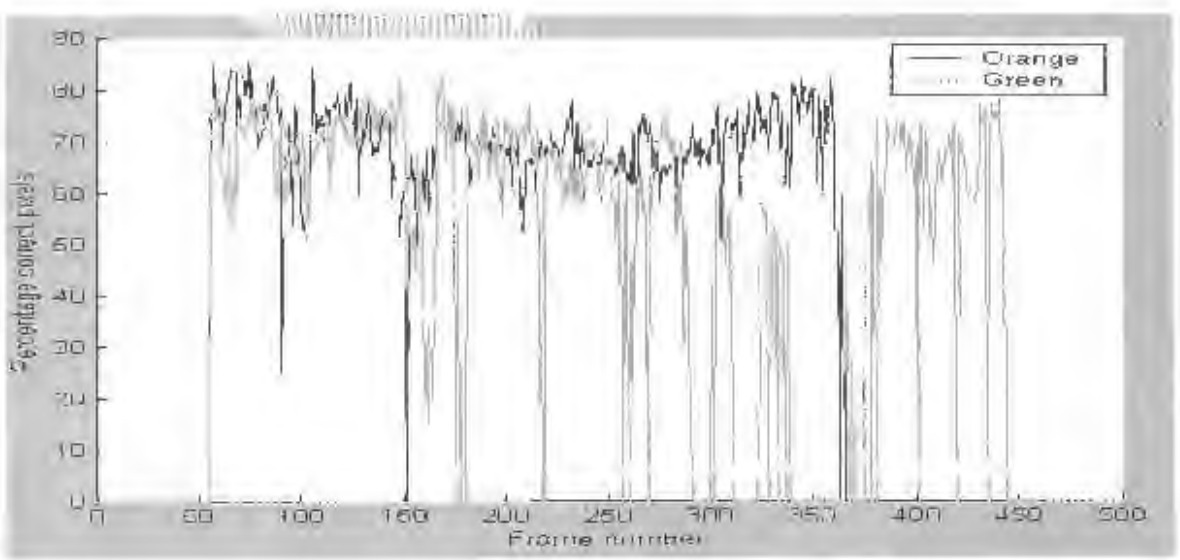

Figure 7.6: The average of the percentages shown in Fig. 7.4 and Fig. 7.5.

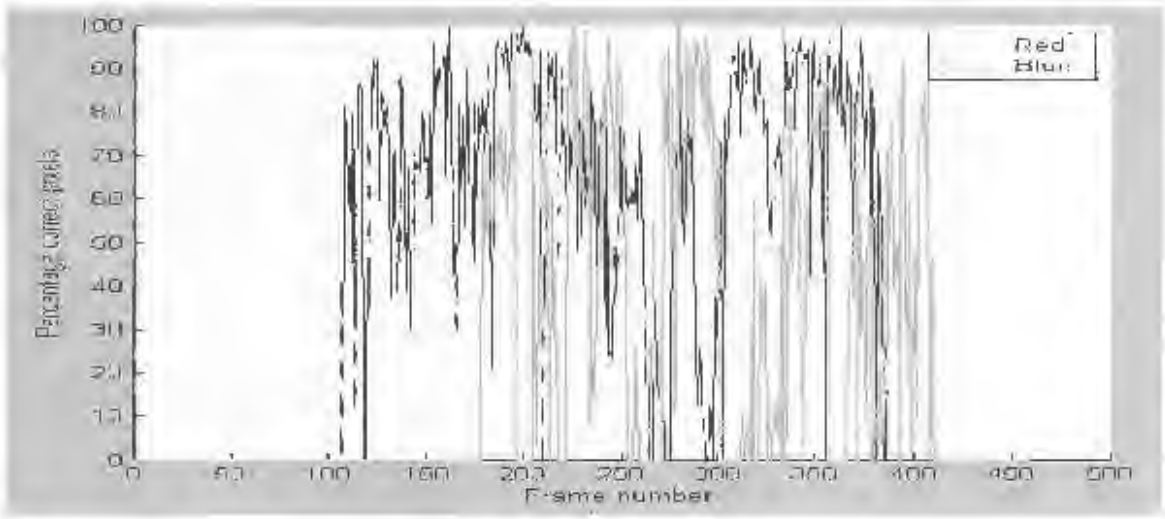

Figure 7.7: Graph of tracking of red and dark-blue targets using the percentage of manually labelled pixels correctly placed in the hypothesis region for each person.

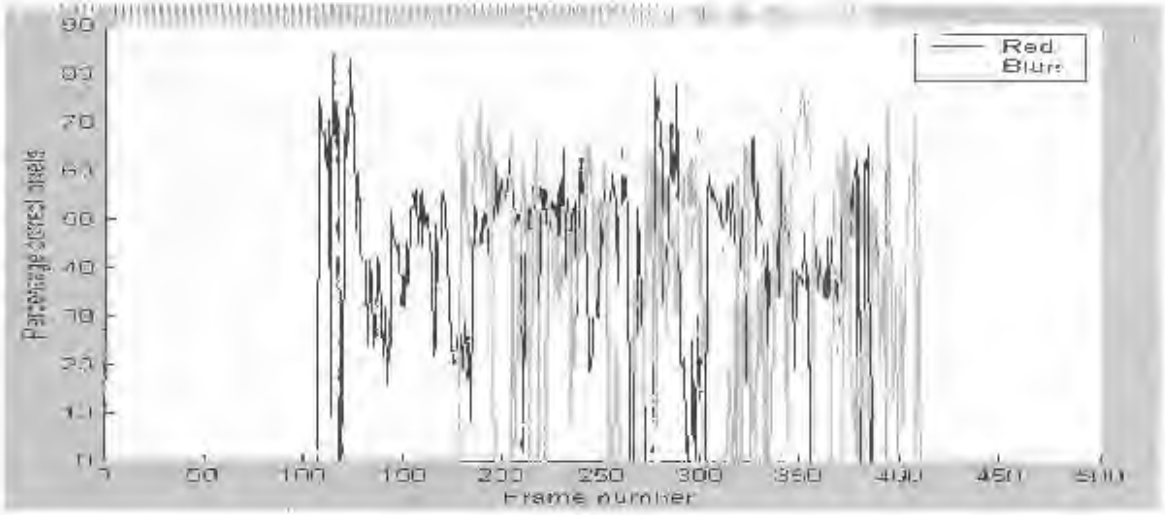

Figure 7.8: Graph of tracking of red and dark-blue targets using the percentage of hypothesized pixels which are correctly labelled as per the manually segmented data.

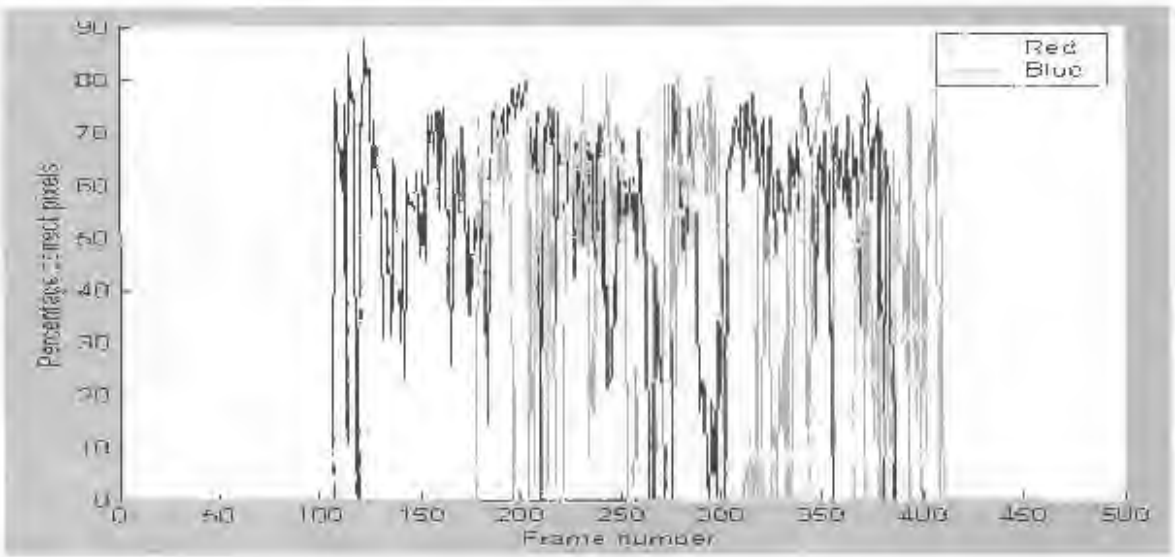

Figure 7.9: The average of the percentages shown in Fig. 7.7, and Fig. 7.8.

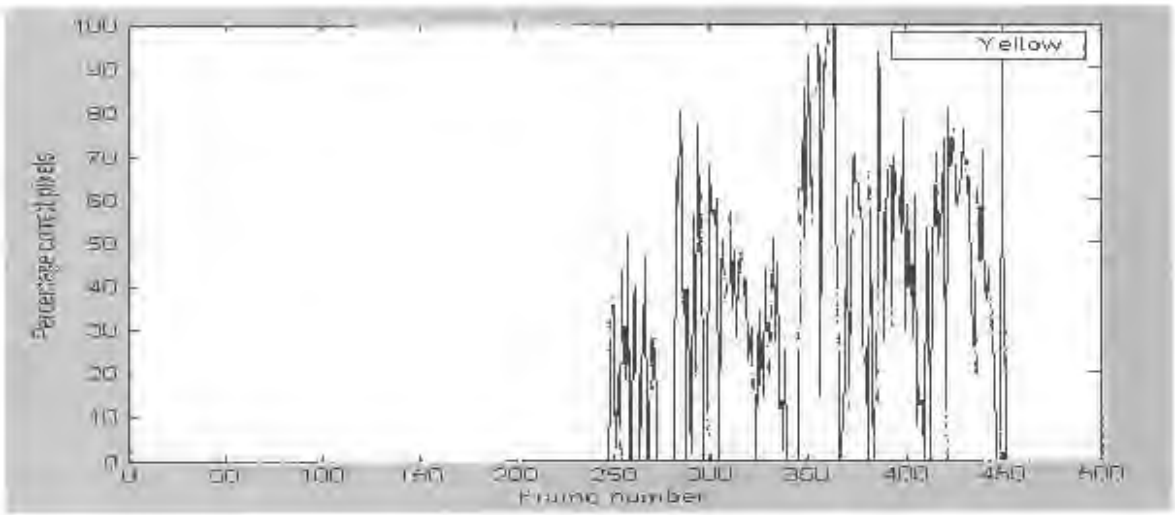

Figure 7.10: Graph of tracking of yellow target using the percentage of manually labelled pixels correctly placed in the hypothesis region for each person.

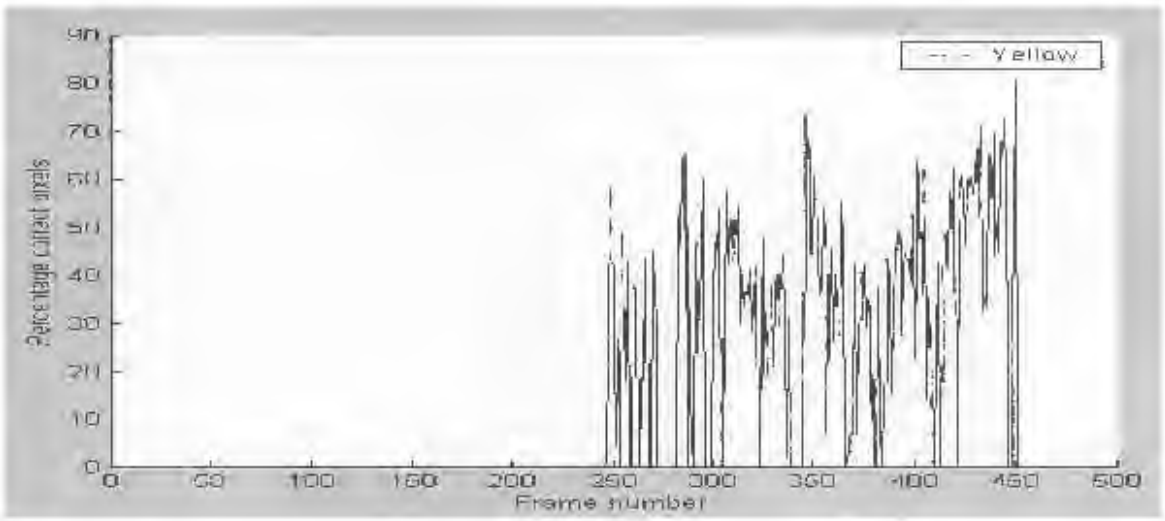

Figure 7.11: Graph of tracking of yellow target using the percentage of hypothesized pixels which are correctly labelled as per the manually segmented data.

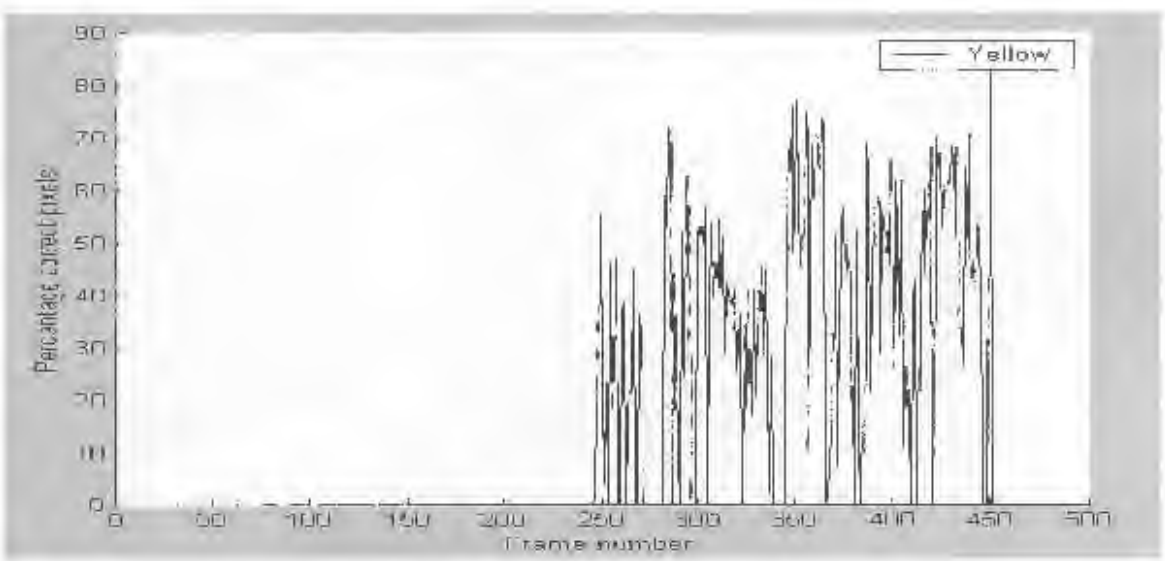

Figure 7.12: The average of the percentages shown in Fig. 7.10, and Fig. 7.11.

	White	L-blue	Orange	Green	Red	D-blue	Yellow
Frames tracked	425	328	308	316	262	165	165
Total frames with target	459	362	310	389	288	236	204
Percentage correct tracks	92.59	90.61	99.35	81.23	90.97	69.92	80.88
Longest tracking failure	15	13	1	16	5	10	6

Table 7.1: Table of tracking of seven people over the entire sequence of 460 frames. A frame in which the percentage of correct pixel labelling is higher than zero is counted as a correct track for that target for that frame.

these graphs is that in general the characteristic of the graphs in Fig. 7.1, Fig. 7.4, Fig. 7.7 and Fig. 7.10 is that the percentage correct pixel classification is higher than in Fig. 7.2, Fig. 7.5, Fig. 7.8 and Fig. 7.11. This is a reflection of the fact that an ellipsoid is not a perfect model of the various image projections of the human figure. The ellipsoid tends to contain, on a correct hypothesis, the torso and legs of the person, but often limbs protrude from the border of the hypothesis region.

Important information about the tracking of each target which may not be clear in the tracking graphs of the indoor sequence is presented in Table 7.1

In this table, the "Frames Tracked" measure is of the number of frames in the video sequence in which some part of the particular target was correctly tracked. The "Total frames with target" parameter indicates how many frames in the sequence the target was present at all. The "Longest tracking failure" parameter indicates the longest continuous number of frames during which the object was lost.

Another run of the algorithm on the same data

Next we display precisely the same information, using the same tracking algorithm run on the same sequence at a different time. These tracking graphs are shown in Figures 7.13 to 7.24. The goal is to compare the tracking reliability if all other factors such as the image data and initialisation parameters are held constant.

We can see that there are some slight differences in the tracking in the first and second runs. For example we can see that in the first run the orange target (Fig. 7.4) was lost around frame 150, but this does not happen in the second tracking sequence (Fig. 7.16). Similarly the white target was lost at around frame 80 in the second sequence (Fig. 7.13), but in the first sequence (Fig. 7.1) it was not. We can see that the red target was lost in the first sequence around frame 110 (Fig. 7.7), but was not lost in the second sequence

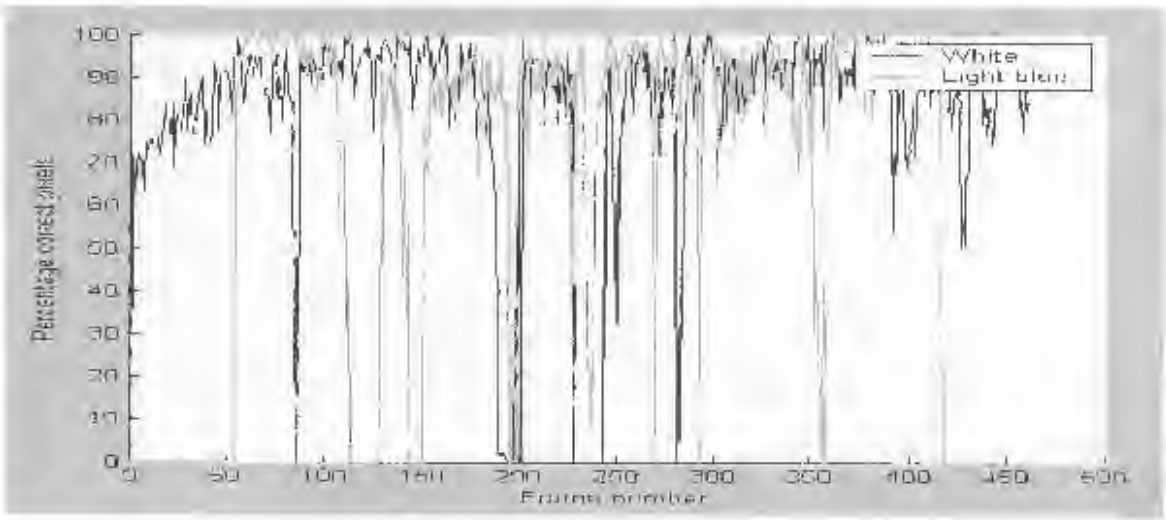

Figure 7.13: Graph of tracking of white and light-blue targets using the percentage of manually labelled pixels correctly placed in the hypothesis region for each person, in the second run on the benchmark sequence.

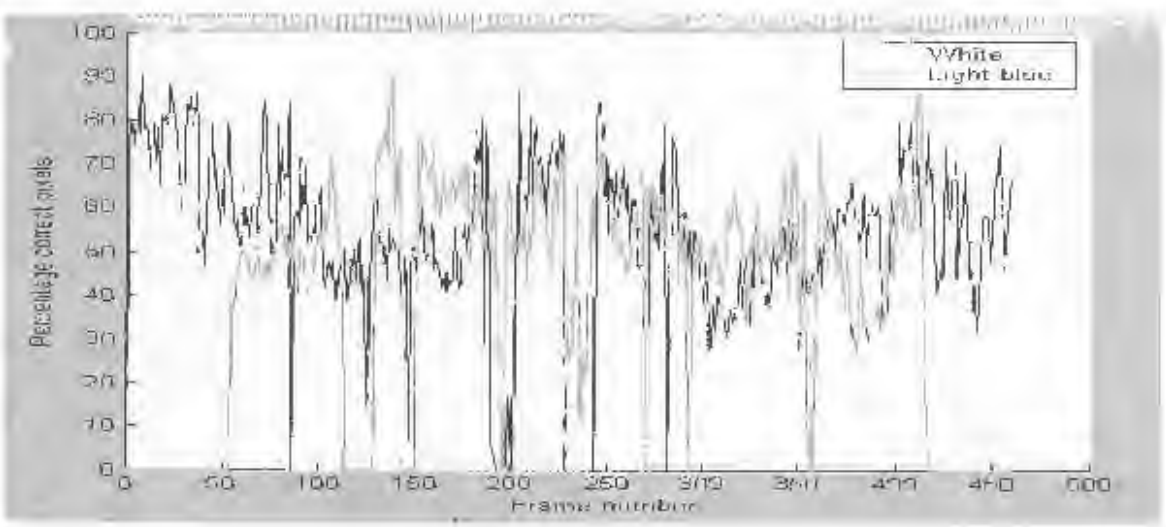

Figure 7.14: Graph of tracking of white and light-blue targets using the percentage of hypothesized pixels which are correctly labelled as per the manually segmented data, in the second run on the benchmark sequence.

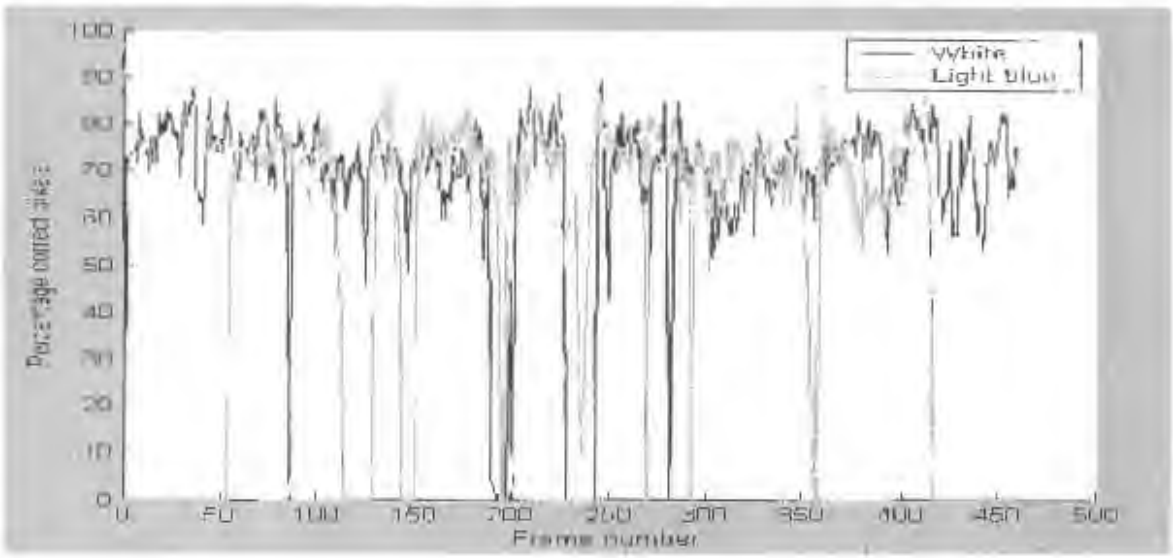

Figure 7.15: The average of the percentages shown in Fig. 7.13, and Fig. 7.14.

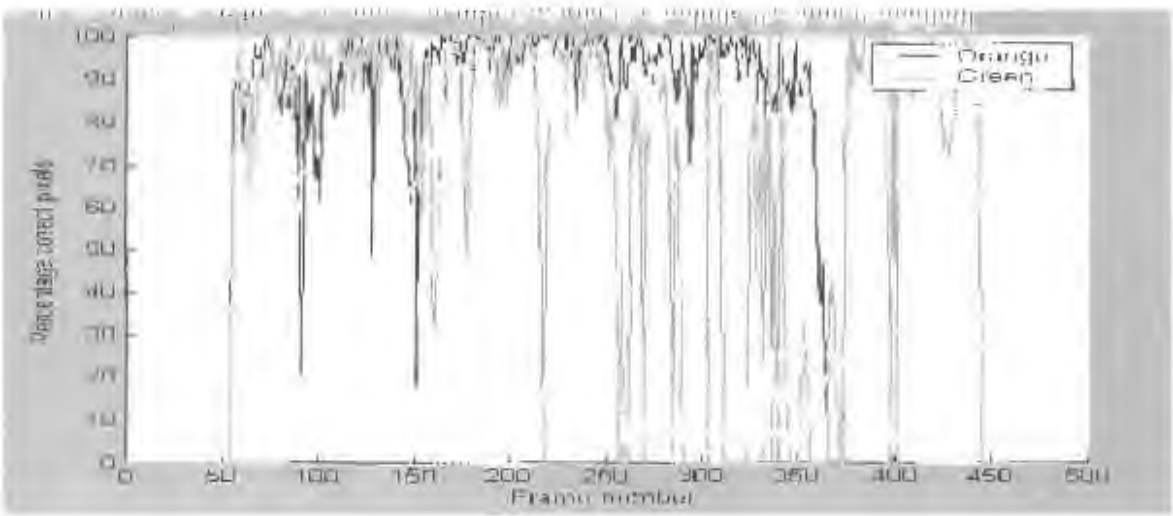

Figure 7.16: Graph of tracking of orange and green targets using the percentage of manually labelled pixels correctly placed in the hypothesis region for each person, in the second run on the benchmark sequence.

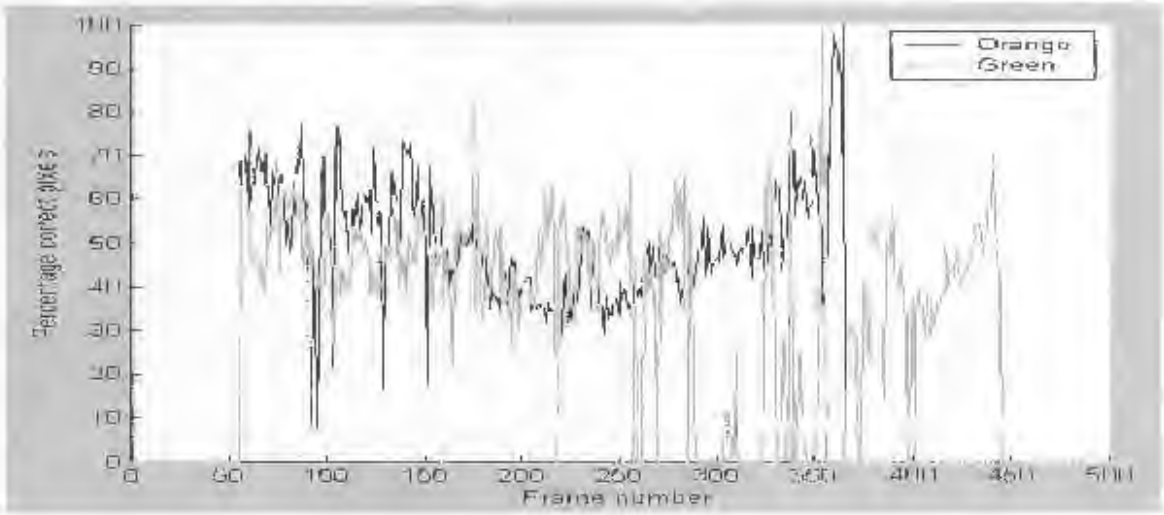

Figure 7.17: Graph of tracking of orange and green targets using the percentage of hypothesized pixels which are correctly labelled as per the manually segmented data, in the second run on the benchmark sequence.

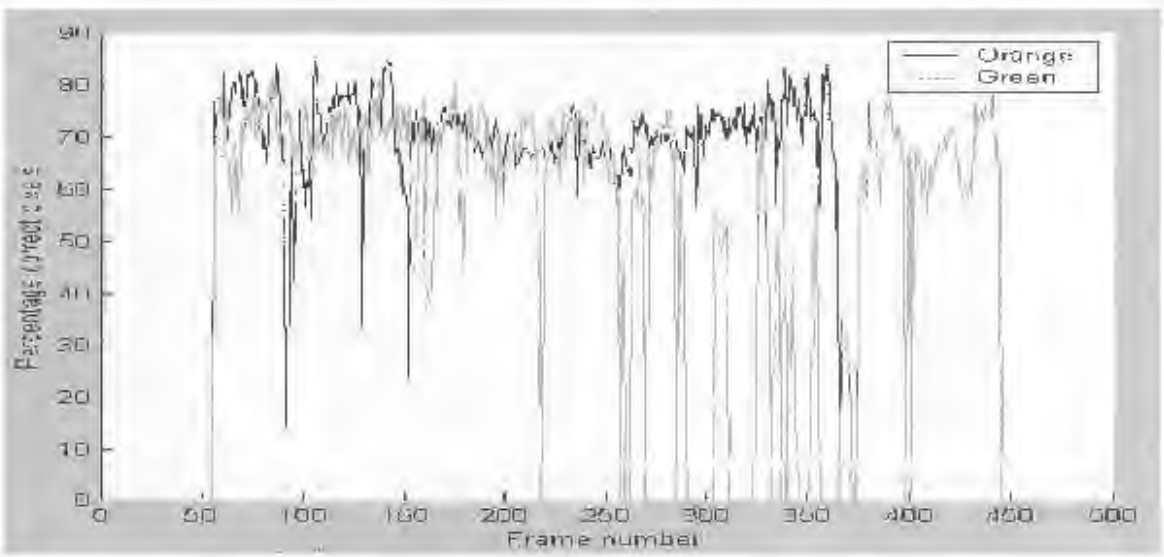

Figure 7.18: The average of the percentages shown in Fig. 7.16, and Fig. 7.17

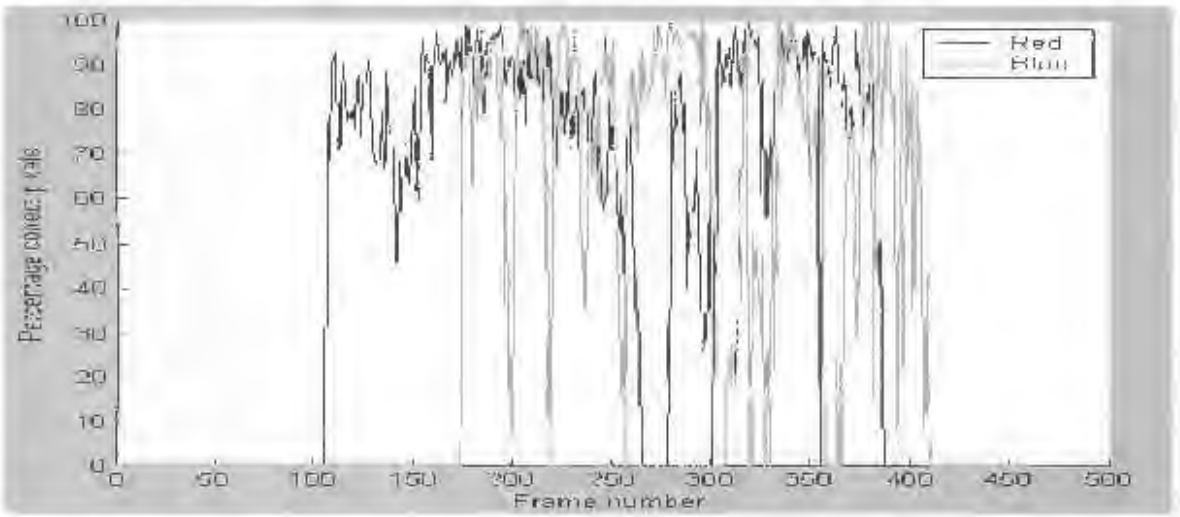

Figure 7.19: Graph of tracking of red and dark-blue targets using the percentage of manually labelled pixels correctly placed in the hypothesis region for each person, in the second run on the benchmark sequence.

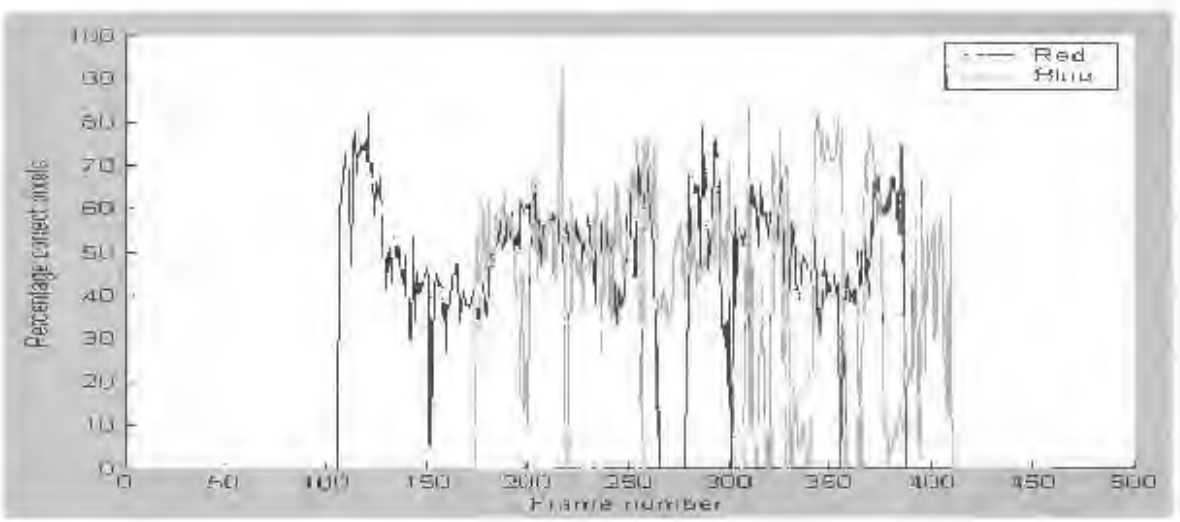

Figure 7.20: Graph of tracking of red and dark-blue targets using the percentage of hypothesized pixels which are correctly labelled as per the manually segmented data, in the second run on the benchmark sequence.

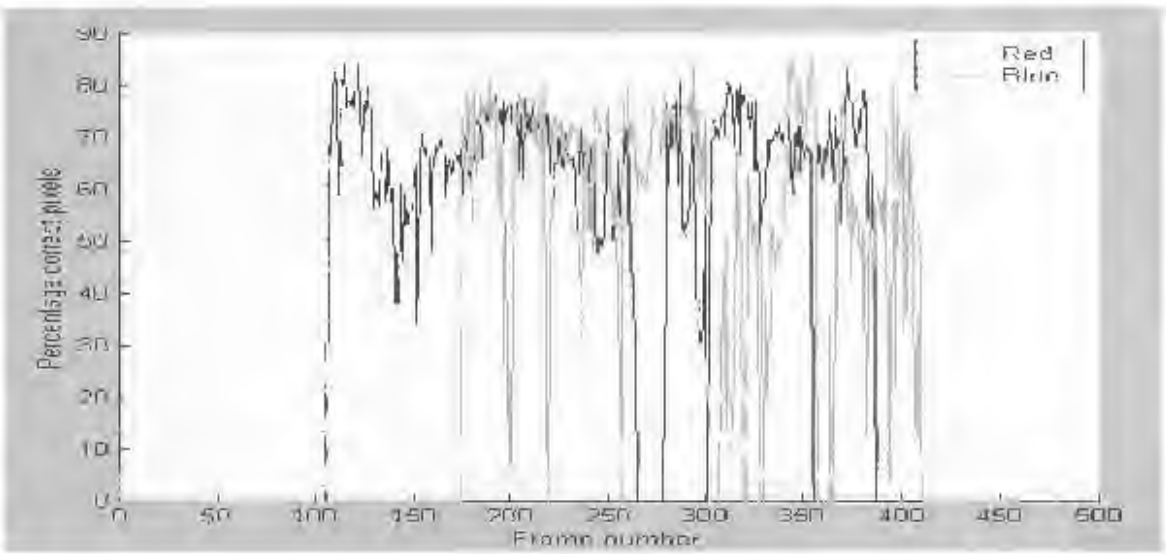

Figure 7.21: The average of the percentages shown in Fig. 7.19, and Fig. 7.20.

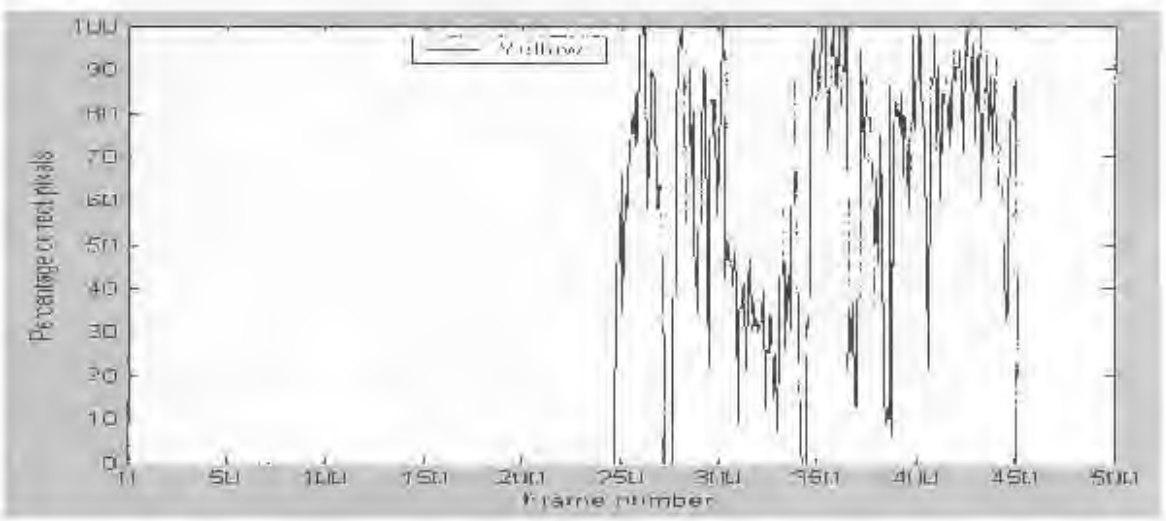

Figure 7.22: Graph of tracking of yellow target using the percentage of manually labelled pixels correctly placed in the hypothesis region for each person, in the second run on the benchmark sequence.

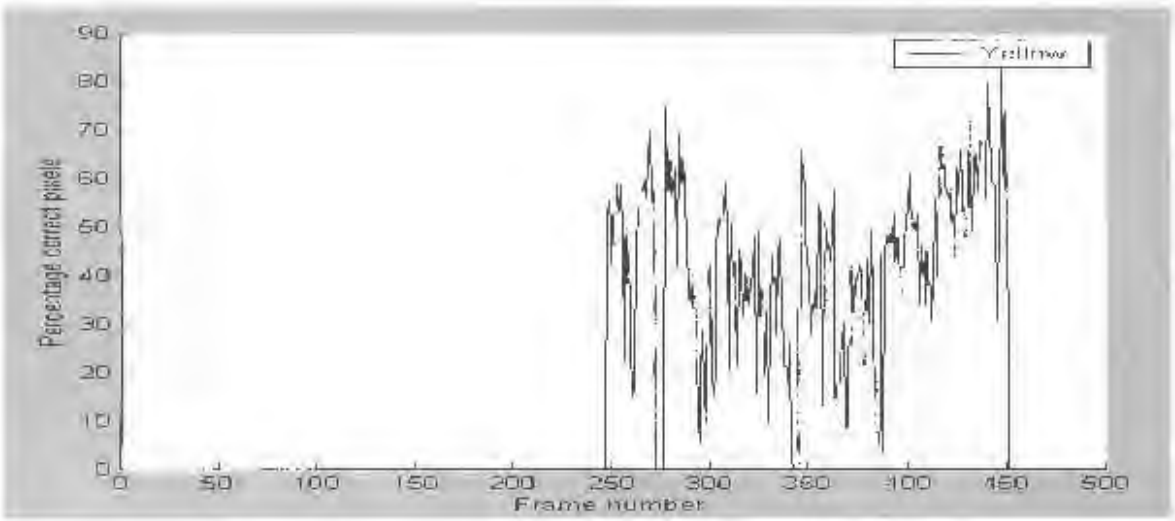

Figure 7.23: Graph of tracking of yellow target using the percentage of hypothesized pixels which are correctly labelled as per the manually segmented data, in the second run on the benchmark sequence.

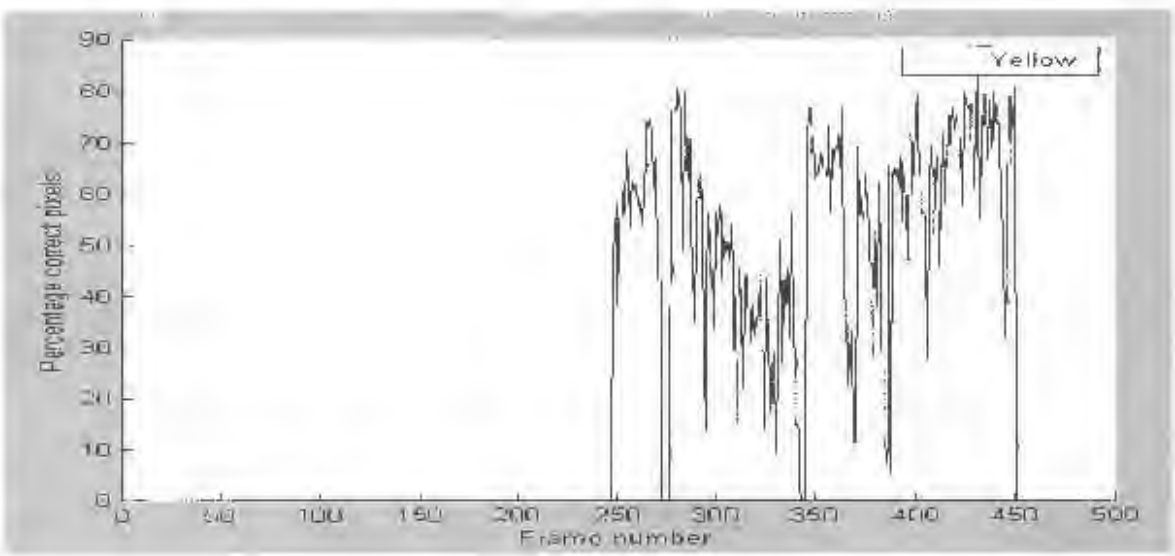

Figure 7.24: The average of the percentages shown in Fig. 7.22, and Fig. 7.23.

	White	L-blue	Orange	Green	Red	D-blue	Yellow
Frames tracked	435	331	310	333	266	215	195
Total frames with target	459	362	310	389	288	236	204
Percentage correct tracks	94.77	91.44	100	85.66	92.36	91.10	95.59
Longest tracking failure	16	17	0	13	6	5	2

Table 7.2: Table of the second run of tracking of seven people over the entire sequence of 460 frames. A frame in which the percentage of correct pixel labelling is higher than zero is counted as a correct track for that target for that frame.

	White	L-blue	Orange	Green	Red	D-blue	Yellow
Frames tracked	125	0	131	116	162	0	41
Total frames with target	459	362	310	389	288	236	204
Percentage correct tracks	27.23	0	42.26	29.82	56.25	0	20.1
Longest tracking failure	278	362	123	83	126	236	161

Table 7.3: Table of tracking of seven people over the entire sequence of 460 frames, using the basic particle filter with 200 particles. A frame in which the percentage of correct pixel labelling is higher than zero is counted as a correct track for that target for that frame.

(Fig. 7.19). The statistical information about the tracking of each target in the second run is presented in Table 7.2.

The important point is that the overall characteristic of the tracker is fairly constant, and from this tracking visualization we can make inferences about its stability, reliability and suitability for a particular application.

7.1.2 Performance of basic particle filter

We now examine the same graphically represented results on the performance of the basic particle filter, using the same joint state and observation model, when applied to the same video sequence using the same initialization data. They are displayed in Fig. 7.25 and Fig. 7.26. The statistical information about the tracking of each target in the run of the basic particle filter is presented in Table 7.3.

The basic particle filter performs poorly compared to the partitioned particle filter. The algorithm here used the same number of particles as did the partitioned particle filter in the

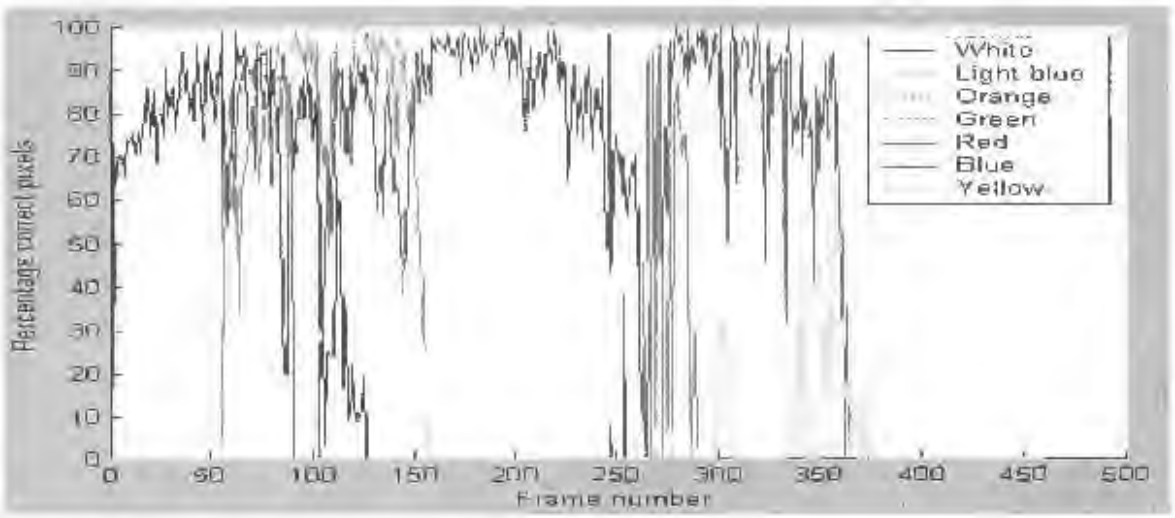

Figure 7.25: Graph of tracking of seven people using the percentage of manually labelled pixels correctly placed in the hypothesis region for each person, using the basic particle filter, with 200 particles.

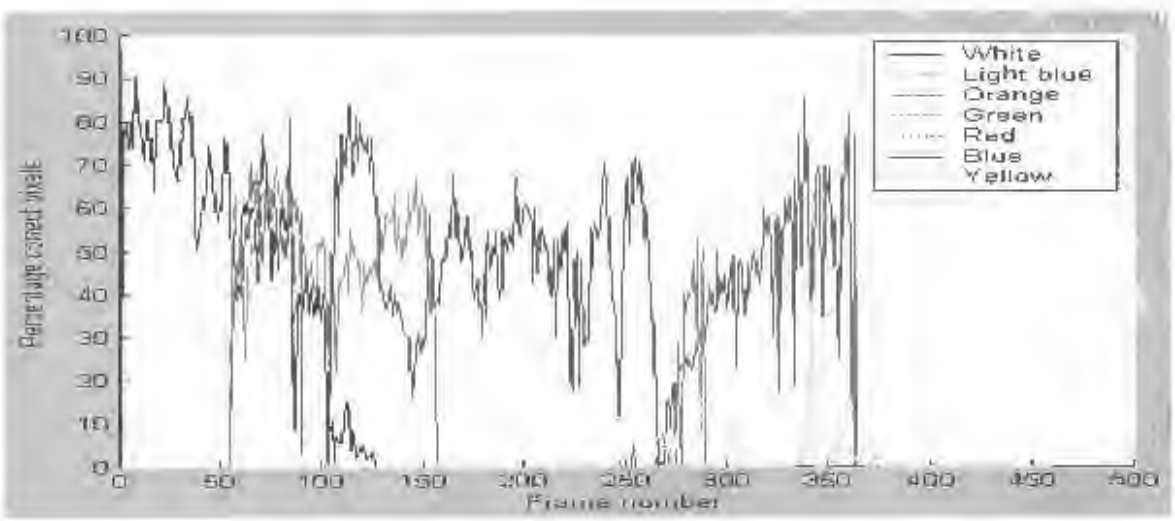

Figure 7.26: Graph of tracking of seven people using the percentage of hypothesized pixels which are correctly labelled as per the manually segmented data, using the basic particle filter, with 200 particles.

	White	L-blue	Orange	Green	Red	D-blue	Yellow
Frames tracked	79	189	2	256	285	0	0
Total frames with target	459	362	310	389	288	236	204
Percentage correct tracks	17.21	52.21	0.65	65.81	98.96	0	0
Longest tracking failure	148	101	308	131	2	236	204

Table 7.4: Table of tracking of seven people over the entire sequence of 460 frames, using the basic particle filter, with 1400 particles. A frame in which the percentage of correct pixel labelling is higher than zero is counted as a correct track for that target for that frame.

graphs shown in Figures 7.1 to 7.24, in the previous subsection. We can see that the basic particle filter is only really capable of tracking a single person at a time, and although in principle the basic particle filter should converge to the same posterior distribution as the partitioned particle filter (although not the same posterior as the partitioned particle filter with scan phase), due to the lack of intelligent sampling, many more particles are needed to perform the same tracking operation.

When the target is lost and then reacquired, as in the case of the green and orange target object, this is purely accidental, and corresponds to a person walking randomly through the predicted region of hypothesis for that person. This aspect of the tracker is entirely unreliable, and so the tracking performance should actually be regarded as worse than is suggested by the tracking graphs in Fig. 7.25 and Fig. 7.26.

It can be argued that if we are to make a performance based comparison between a basic particle filter and a partitioned particle filter, then it is the relative amount of computation per frame which should be held constant in each case, rather than the number of particles. Since with the partitioned particle filter each particle is processed once per partition, we could increase the number of particles in the basic particle filter by a multiplicative factor equal to the number of partitions. This would cause the computational expense between the two algorithms, per frame, to be equal. The performance of the basic particle filter with 1400 particles, (since the number of particles per partition in the partitioned particle filter is 200, and there are seven partitions), is depicted in Fig. 7.27, Fig. 7.28 and shown in Table 7.4. The performance is slightly better than that of the basic particle filter using 200 particles: the filter can for short periods track three targets, but not more than this. At this point the advantage of using partitioned sampling becomes clear. In theory, as the number of particles tends to infinity, both the basic particle filter and the partitioned particle filter will converge to the same posterior. We can see only a small improvement in the basic particle filter, although we increased the number of particles by a factor of seven.

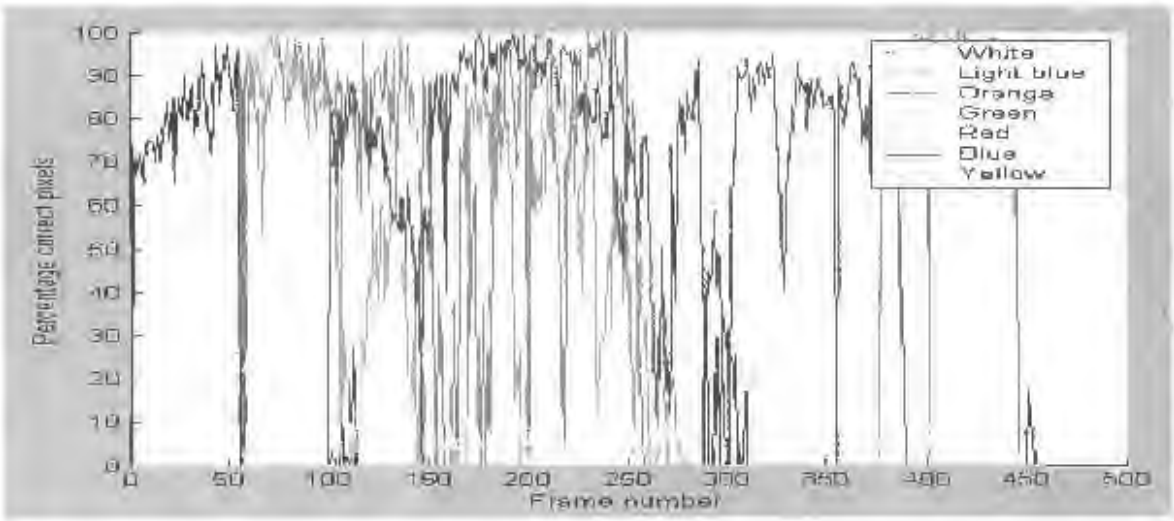

Figure 7.27: Graph of tracking of seven people using the percentage of manually labelled pixels correctly placed in the hypothesis region for each person, using the basic particle filter with 1400 particles.

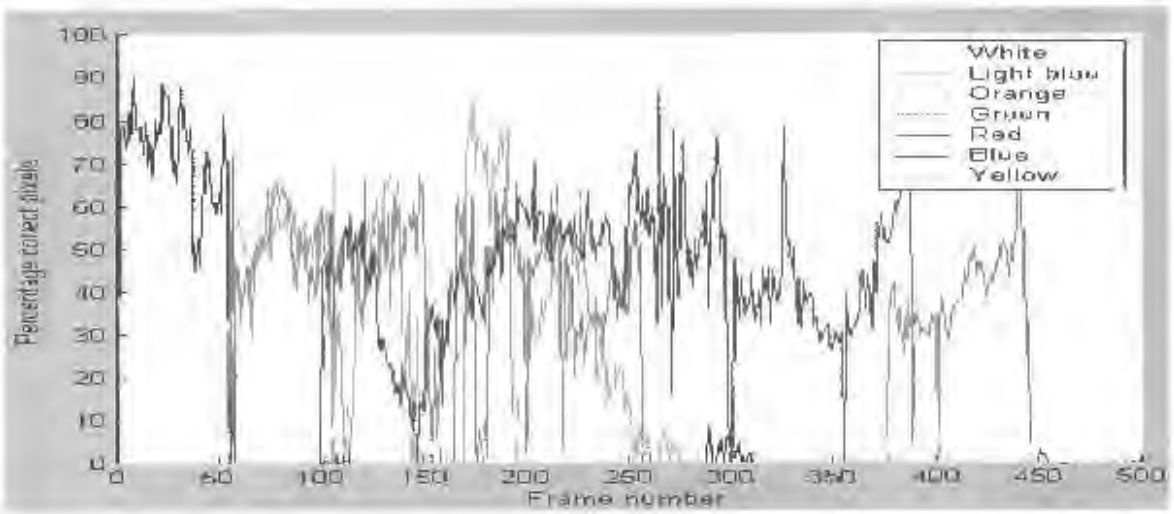

Figure 7.28: Graph of tracking of seven people using the percentage of hypothesized pixels which are correctly labelled as per the manually segmented data, using the basic particle filter with 1400 particles

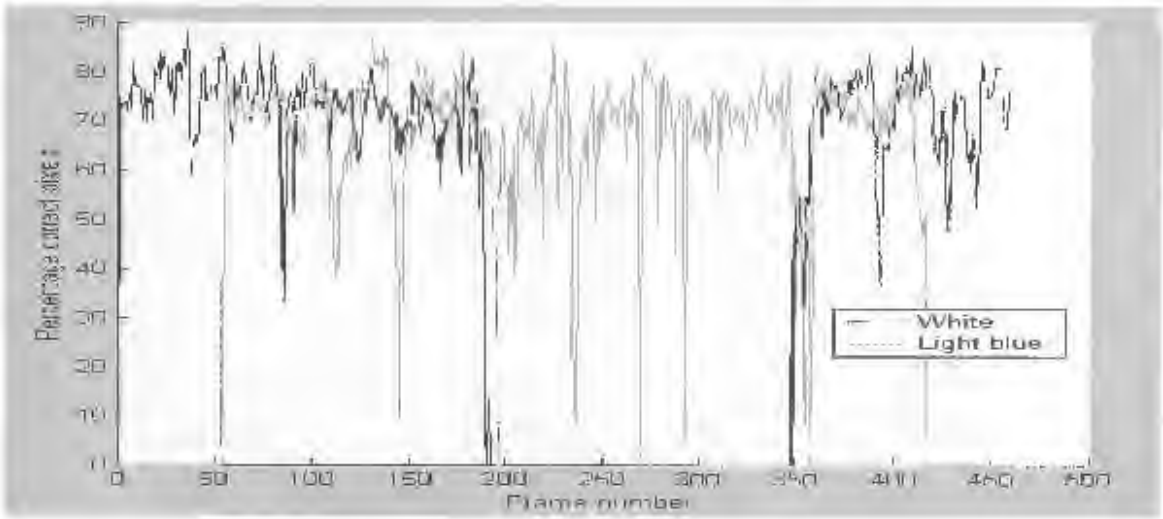

Figure 7.29: Graph of tracking of seven people using the percentage of hypothesized pixels which are correctly labelled as per the manually segmented data, using the partitioned particle filter without the scan phase.

7.1.3 Performance of partitioned sampling algorithm without scan phase

It is instructive to examine the performance using the pixel percentage metric of the partitioned particle filter implemented without the scan phase. The results are as shown in Fig. 7.29, Fig. 7.30 and Fig. 7.31. The statistical information about the tracking of each target in the run of the partitioned particle filter without the scan phase is presented in Table 7.5.

	White	L-blue	Orange	Green	Red	D-blue	Yellow
Frames tracked	302	358	309	247	227	213	184
Total frames with target	459	362	310	389	288	236	204
Percentage correct tracks	65.8	98.9	99.68	63.5	78.82	90.25	90.2
Longest tracking failure	151	1	1	132	53	6	8

Table 7.5: Table of tracking of seven people over the entire sequence of 460 frames, using the partitioned particle filter without the scan phase. A frame in which the percentage of correct pixel labelling is higher than zero is counted as a correct track for that target for that frame.

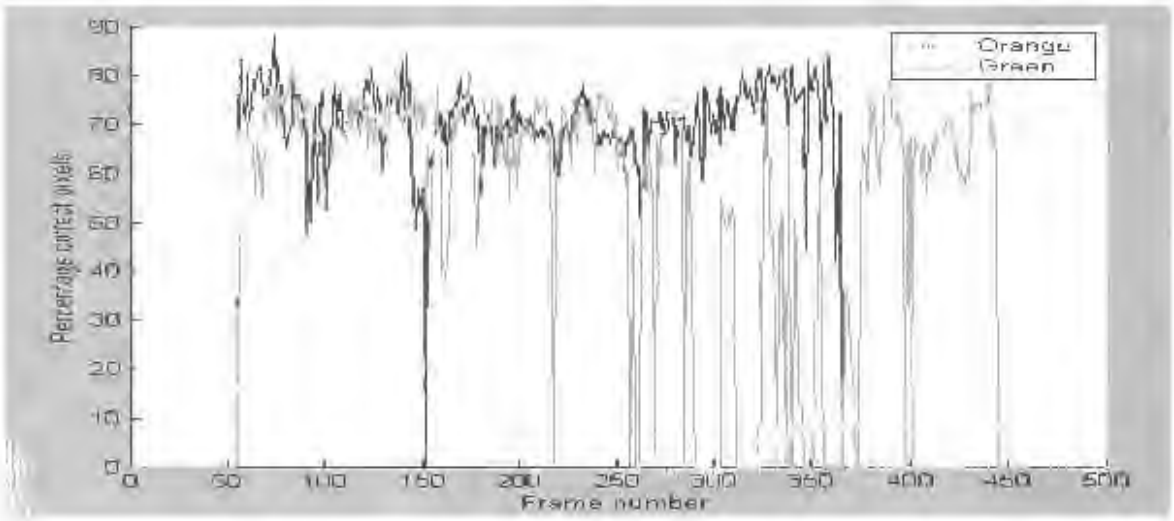

Figure 7.30: Graph of tracking of seven people using the percentage of hypothesized pixels which are correctly labelled as per the manually segmented data, using the partitioned particle filter without the scan phase.

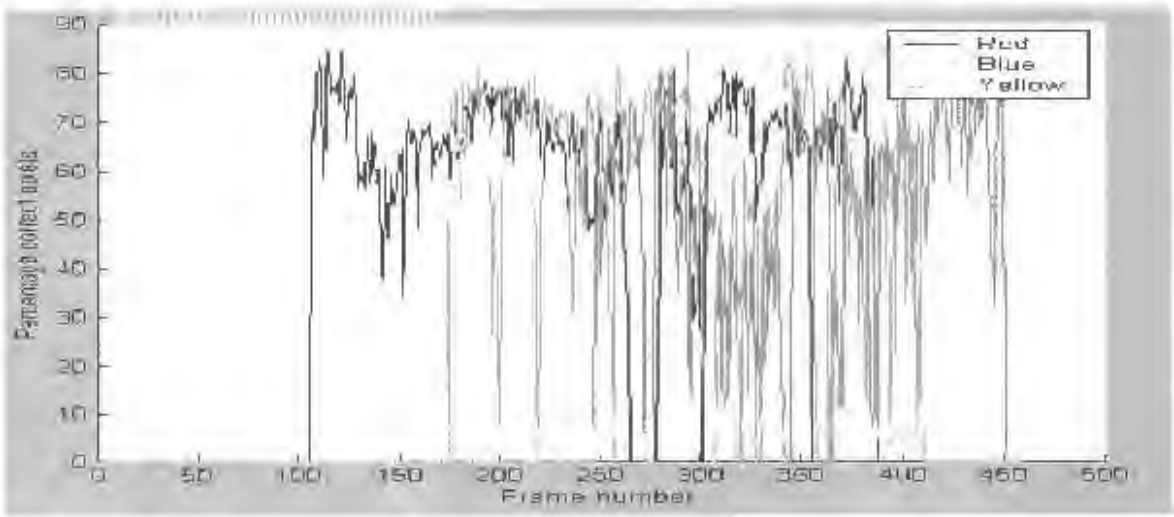

Figure 7.31: Graph of tracking of seven people using the percentage of hypothesized pixels which are correctly labelled as per the manually segmented data, using the partitioned particle filter without the scan phase.

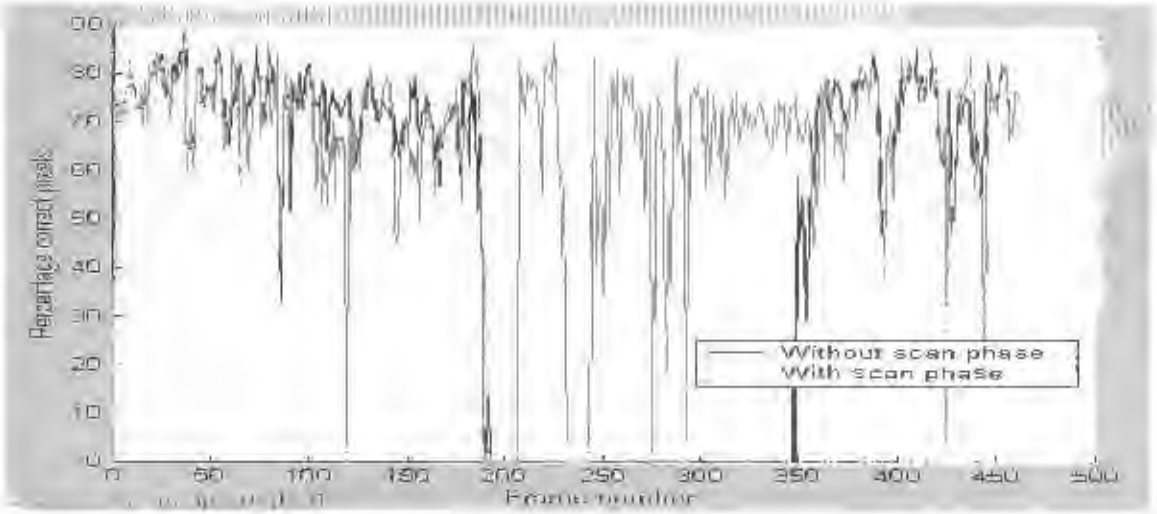

Figure 7.32: Graph of tracking of white target using the average percentage metric, using the partitioned particle filter without the scan phase.

The graphs of the tracking performance in Fig. 7.29, Fig. 7.30 and Fig. 7.31 are not particularly revealing, as the general characteristics look very similar to that of the data for the partitioned particle filter with the scan phase. We can see in the Table 7.5 however that the figure for the "Longest tracking failure" has increased drastically, and this is because without the scan phase, a lost target can only be recovered accidentally. We can gain a greater understanding of the benefit of the scan phase if we examine the tracking plot for a particular individual, with and without the additional scan phase. We can see two such graphs, in Fig. 7.32 and Fig. 7.33.

We can see in the tracking information shown in Fig. 7.32 more clearly how the white target is completely lost and then accidentally regained in the case of the partitioned particle filter without the scan phase. If we add the scan phase, the white target is recovered quickly after occlusions.

Fig. 7.33 is a similar plot, but this time showing the tracking information of the red target. Once again the filter performs better if the scan phase is used, since the target is lost at around frame 330 without the scan phase, but is tracked until frame 385 if the scan phase is included. We may observe here also that in fact the scan phase introduces more false hypotheses than would otherwise be processed, so in fact there is also a negative impact on the performance: At several occasions, (frames 120, 210, 260, 355) the filter with

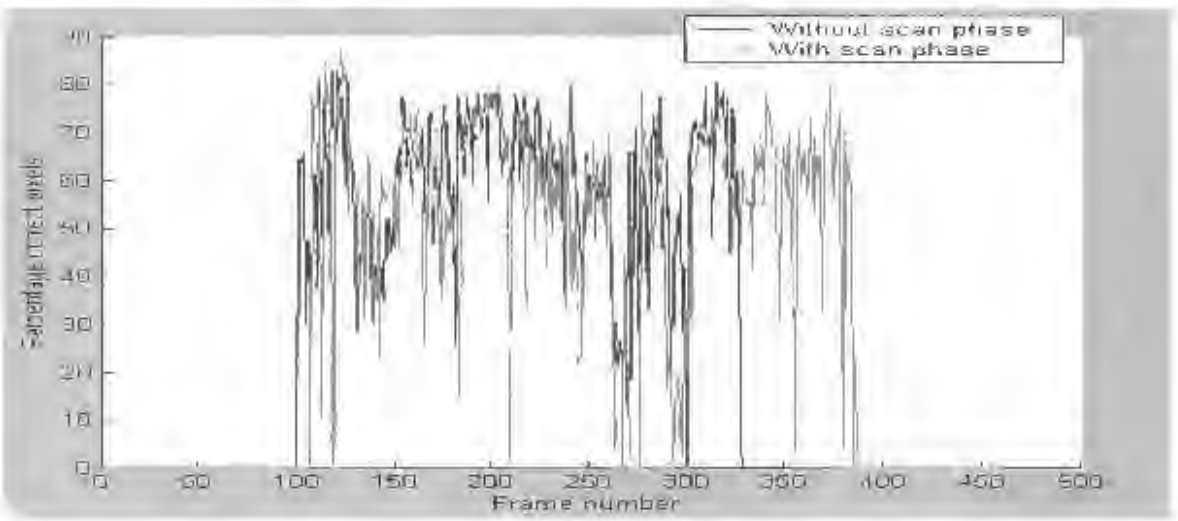

Figure 7.33: Graph of tracking of red target using the average percentage metric, using the partitioned particle filter without the scan phase.

the scan phase temporarily loses the target for short periods (one or two frames), where the partitioned particle filter without the scan phase does not. This is because during short periods of near total occlusion which the filter without the scan phase might survive, the filter with the scan phase would more easily lock on to an alternative hypothesis elsewhere in the image. The scan phase allows the filter to recover quickly from its scan phase induced error.

7.2 Effective Number of Particles per Partition

Another way of measuring the effectiveness in the sampling/dynamics stage of the selection of samples from an appropriate region in the state space, is to use Doucet's estimated effective number of particles [68], proposed in 1998. Where the effective number of particles, N_{eff}, is calculated as:

$$N_{\text{eff}} = \left(\sum_{i=1}^{N} \pi_i^2 \right)^{-1}$$

Intuitively this measure indicates the number of useful particles which may be contained in the distribution containing these weights, in the sense that particles with very low probabilities are not likely to be used again, and so are wasted time and space.

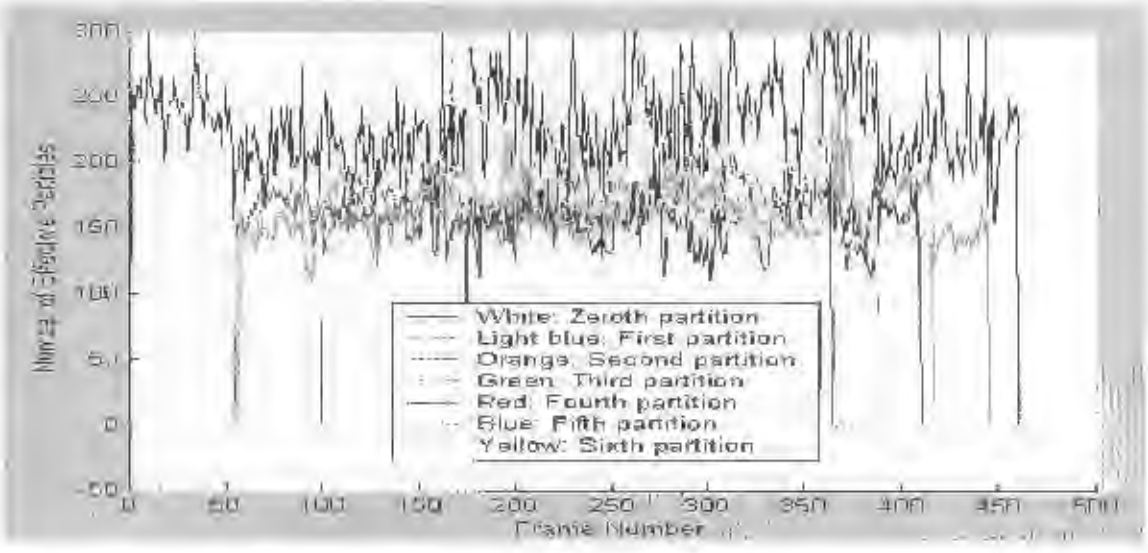

Figure 7.34: The Number of Effective Particles shown for each partition, for each time step.

If all the particles have equal weight $1/N$, then $N_{\text{eff}} = N$, which shows that all the particles are likely to be used. Conversely, if the variance of the particles is very high, then $N_{\text{eff}} = 1$.

The plot shown in Fig. 7.34 shows the number of effective particles calculated at each partition, after the importance reweighting has been applied, but before the resampling step has taken place, for the indoor sequence. Since each partition is associated with one of the target objects, we notice that certain partitions only become active as the corresponding target appears on the scene. The number N_{eff} is highest for the zeroth partition, corresponding to the white person, and lower for subsequent partitions. The reason for this is that in this implementation the N_{eff} for the zeroth partition in fact corresponds to the posterior for the previous time step. In a successful partitioned particle filter, we expect the N_{eff} of the posterior distribution for any time step in general to be higher than the N_{eff} for the iterative searches through the partitions that would lead up to such a posterior.

Notice that while the data shown in Fig. 7.34 gives us an idea of when the various partitions become active, and perhaps also of how many of the samples at each partition are in fact being used, the N_{eff} measure presented for a particular, typical tracking sequence does not correlate usefully with the percentile match metrics shown in Figures 7.1 to 7.12, which provide ground-truth data for the tracking success at any time instant for the first

run on the benchmark sequence. The value of N_{eff} as a tracking measure is then cast into question, although in general it may still be useful to determine the tracking efficiency, if it is already known that the entire tracking sequence was successful, or if it is known which parts were successful.

7.3 State space variance per dimension per time-step

The primary contribution of particle filtering in state space estimation is to allow for the representation and propagation of multi-modal state estimation given multi-modal process and observation noise. Therefore, the variance about the expectation value is not necessarily a good measure for the successful tracking of an object. However, as with N_{eff}, under certain assumptions it can be an indicator for the efficiency of the filter. If the true state space distribution is in fact unimodal, as would be the case if each target object were uniquely identifiable, and if the posterior distribution $p(\mathbf{X}|\mathbf{Z})$ of this state space is essentially unimodal, and if it is unimodal about the correct mode (i.e. the tracking is successful), then the variance gives an indication of the accuracy to which the correct solution has been found, and also of the efficiency of the filter.

Presented in Fig. 7.35 is the variance of the posterior distribution at each time step (to give the variance of each dimension at each time step for each partition is not useful), for the indoor scene.

We may have expected some correlation between the variances within each dimension and the corresponding ground-truth of the quality of the tracking for the objects as shown in Figures 7.1 to 7.12. However this is not the case, and the two statistics seem unrelated to one another. One may ask why there is not a large increase in the variance of a particular object's location vector when that object is lost in the tracking sequence, occluded or otherwise. Although there may be a small unnoticed correlation in this regard, the algorithm usually recovers from lost targets within one to three frames, and this is typically not enough for the variance to build up noticeably. An alternative explanation is that the scan phase of the algorithm is both its strength and its weakness. When an object is lost, an alternative, although incorrect hypothesis is usually found very quickly. Local maxima for the posterior probability would still exist in configurations where the lost target is incorrectly hypothesized to be near some image data which best approximates it, and the characteristic of these local maxima is that they are sufficiently high to trap particles and therefore maintain them at a low variance within a particular subspace in the state space.

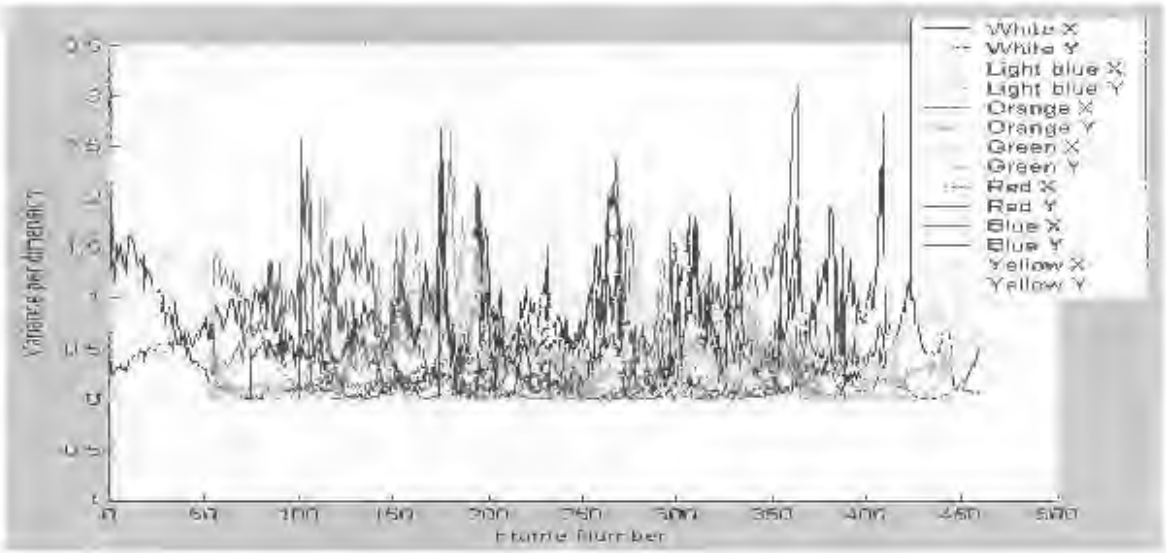

Figure 7.35: The variance within each dimension at the posterior for each time step.

	First	Second	Third	Fourth
Frames tracked	144	139	150	147
Total frames with target present	153	153	153	153
Percentage correct tracks	94.12	90.85	98.04	96.08
Longest tracking failure (frames)	5	8	1	2

Table 7.6: Tracking information for the first natural outdoor scene with four targets.

7.4 Tracking results for outdoor scenes

We now examine some of the results for the tracking of the outdoor sequences. We see in Fig. 7.36 the average correct pixel classification per frame, for the first outdoor sequence, and in Fig. 7.37 the statistics for the second outdoor sequence.

We see in 7.36 that the the performance for the first outdoor sequence is reasonably good, and that targets were momentarily lost, but that the tracking overall was sound. Table 7.6 summarizes the information in Fig. 7.36.

Similarly, in Fig. 7.37 we see that the the performance for the second outdoor sequence is also good, and that targets were momentarily lost, but that the tracking overall was sound.

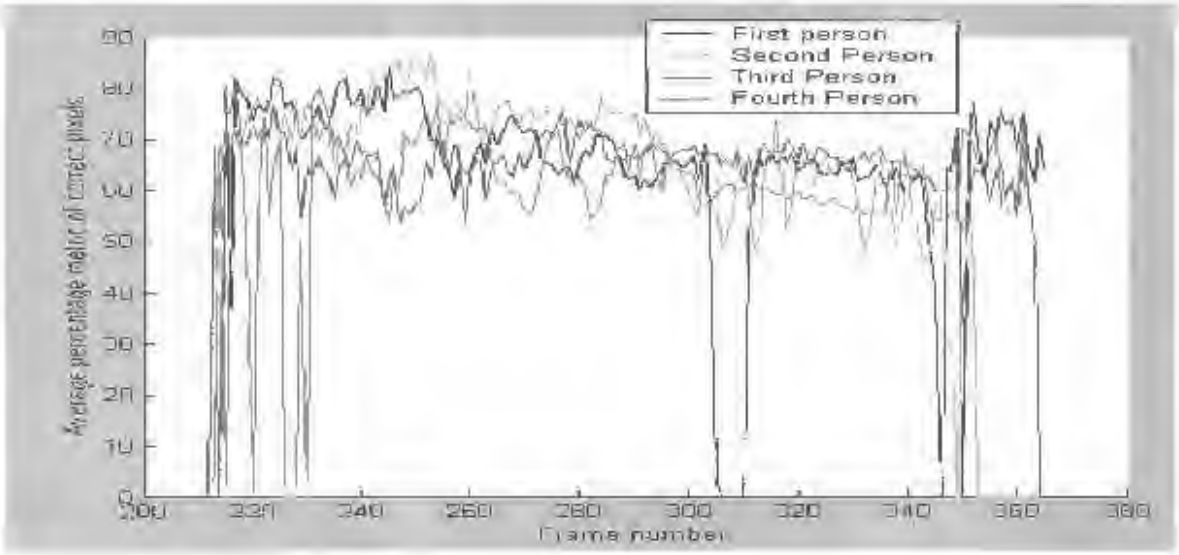

Figure 7.36: The average of the two percentile metrics, for the first outdoor scene in which there are four people, dressed naturally, with simple interaction.

Table 7.7 summarizes the information in Fig. 7.37.

7.5 Operating speed of tracking algorithm

As mentioned in the introduction, this implementation of the algorithm was not optimized for speed, but rather for extensibility. Moreover, the most expensive stage of the algorithm, which is the observation stage, requires the iterative processing of the current image, once per particle, at each partition. The partitioned sampling does result in an overall speed increase through intelligent selection of the state space region in which to sample, but

	First	Second	Third	Fourth
Frames tracked	145	142	93	116
Total frames with target present	147	147	110	147
Percentage correct tracks	98.64	96.60	84.55	78.91
Longest tracking failure (frames)	2	1	10	6

Table 7.7: Tracking information for the second natural outdoor scene with four targets.

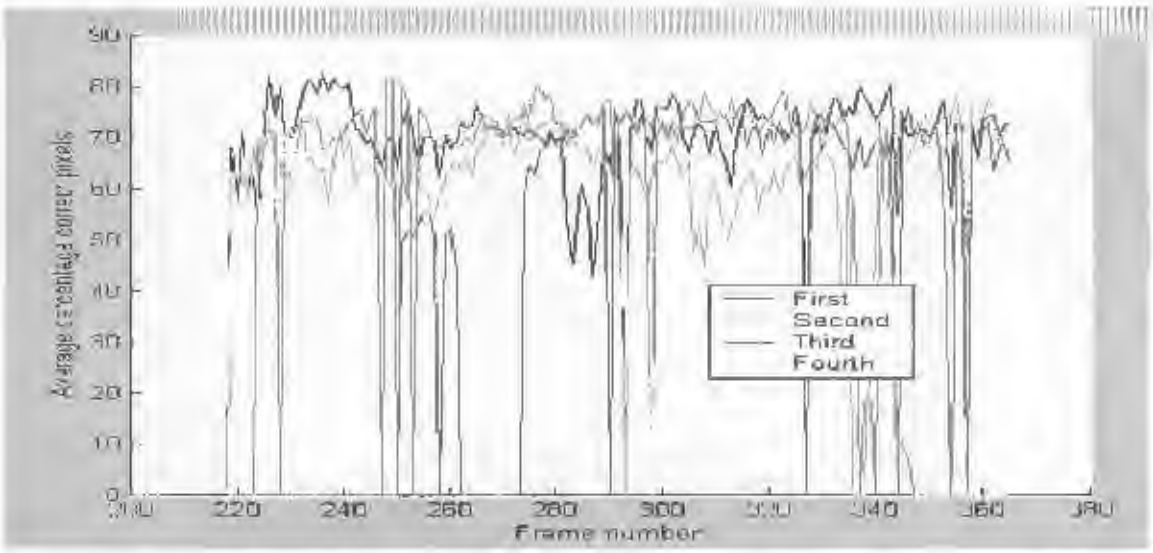

Figure 7.37: The average of the two percentile metrics, for the second outdoor scene in which there are four people, dressed naturally, with simple interaction.

the processing overhead is too high to allow for real-time tracking despite this particular optimization.

As in [1], the pixel classification stage is moved into a preprocessing stage, so that the probability maps are already available for the particle filter to operate on. This simplifies the observation process, since each particle then needs only to sum over the probability map (in which probabilities are represented in the log domain) for that particle for that image in the appropriate way. A single frame in the video sequence takes approximately two to three minutes to process, based on one partition per person, with seven people, with two hundred base particles and one hundred augmented particles per partition. In many particle filter algorithms, the particle hypothesis is analyzed to extract a region in the image within which to make the observation, however this is not necessarily an appropriate approach, since in that case there is image data which is being ignored, and which may be relevant. With extensive optimization, we may see real-time performance.

To iterate through a probability map once, i.e. to visit each pixel component, and to increment the cumulative observation probability in the log domain takes approximately 3 milliseconds. To draw a sample from the prior distribution, add the noise, and render the ellipsoids (via a scenegraph) which will generate the pixel labelling for a particle into a

bitmap takes approximately 30 milliseconds. The total is then 33 milliseconds per particle. With partitioned sampling, there is an additional overhead of finding the correct order of the partitions whenever a new target arrives or leaves. On this platform (which uses a Celeron 2GHz processor), we can evaluate approximately 30 particles per second. Since we need approximately 300 particles per partition (i.e. per target), and since we have seven targets, we need 2100 particle computations per frame. Assuming that the frame rate for the video feed is 10 frames per second, we require computation for 21000 particles per second, and so the algorithm on this platform is a factor of 700 slower than real-time.

The preprocessing stage for each particle is also very expensive, although it needs to be performed only once per frame. For eight models, with sixteen mixtures per target object model and a unique single centered Gaussian distribution per pixel for the background model, it takes 45 seconds to preprocess a frame to generate probability maps for all targets. This is obviously a large overhead, but the speed of this stage may be increased by reducing the number of mixture centers for the target models (although this will reduce the accuracy of the classification). Also, one may use instead of a full covariance matrix, a spherical covariance matrix, which will greatly reduce the cost of calculating the activation matrix, as per [67]. The cost of using the $L^*a^*b^*$ colour space may also be discarded by using the RGB space, or whichever colour space the data was in when it arrived from the camera.

7.6 Performance of tracker on random initialization

Suppose the tracker is not initialized on the correct data, but rather at random. A measure of how quickly the tracker recovers all the target objects is an indication of the robustness and independence of the tracker from human intervention. It is also relevant to the complete solution tracking algorithm, in which a tracker should be able to detect targets as well as locate them. A tracker which can recover all target objects quickly would also be more independent and useful when integrated with a large network of such trackers, each responsible for a location or area of interest of a building.

The images in Fig. 7.38, Fig. 7.39, Fig. 7.40 and Fig. 7.41 demonstrate how quickly the tracker recovers all the targets, when randomly initialized.

This ability to reacquire targets within a joint hypothesis framework would not be possible with the use of multiple independent trackers, nor without the partitioning which allows the scan phase for individual targets. Any number of targets may be reacquired in this way, provided they are partially visible, and a sufficient number of augmented particles have been allocated to the scan phase.

Figure 7.38: Tracker for four targets is randomly initialized.

Figure 7.39: After one frame, the white and green targets have been recovered.

Figure 7.40: After two frames, the light-blue target has also been recovered.

Figure 7.41: After three frames, each of the four targets has been recovered.

Chapter 8

Conclusion

8.1 Summary of results

In this thesis, experimentation was done in the application of Partitioned Particle Filtering to the tracking of people (target objects) in a RGB video sequence. A variety of colour spaces were tested to investigate the colour separations which they could yield in the colour histograms of the colours of the target objects. It was found that the L*a*b* colour space yielded the best colour separation, and so the Gaussian Mixture Models which were designed to model the colour distributions of the target objects were trained from the data transformed into this space.

The use of Hidden Markov Models built and implemented on a fixed-lag smoothing basis to improve the probability estimates in the pixel classification stage, in theory takes advantage of the underlying and correct assumption that pixels have a low likelihood of making a transition to any other state given the state that they are in. However according to the Log Correct Classification Metric introduced in chapter 4, the overall effect of HMM smoothing is negative given the observation method used in this implementation, although the HMM smoothing could assist in the case of more severe image noise.

The basic particle filtering algorithm was also compared to the partitioned particle filtering algorithm, using the same pixel classification models, and the partitioned particle filtering algorithm was found to perform much better. The traditional particle filter cannot effectively track more than three people in a heavily cluttered scene given a fixed small number of particles, whereas the partitioned particle filter augmented with the scan phase can track all seven of the target objects successfully, although they are lost on numerous occasions, but then recovered via the scan phase.

The scan phase which augments the partitioned particle filter causes an improvement in the performance of the tracker, although it is a relatively inexpensive step to take. At

each partition, the state space is augmented with a small number of altered particles, each of which the tracker evaluates. With each partition in the sample space corresponding to a single target object, this altered set of particles represents samples from the search space which have a corresponding manually defined subspace which the sampling mechanism then explicitly explores.

8.2 Future work

Within the area of the type of observation which is done on samples of the state space, a dynamical observation model may be implemented, such as the adaptive Gaussian Mixture Model. Histogram comparison methods may also be used to compare image data with state hypotheses, as these better allow a region to be compared with a target model, rather than assuming that a sum of pixel classification probabilities yields the same result. The colour models developed for the target objects may be defined on a finer scale, so that the spatial information of the colour distribution on a target object is not lost in the creation of a single Gaussian Mixture Model for the entire object. It may also be beneficial to use multiple colour models for a single target, depending on its orientation to the camera.

Within the partitioned sampling framework, it may be beneficial to allocate the first partition to the target which is most likely to be correctly tracked, and the last partition to the target with the smallest likelihood of being correctly tracked with the same order established for the intermediate partitions (it may be occluded or otherwise), and research may be done on the best way to estimate this likelihood, and of its effects on the tracking performance.

As mentioned previously, we may improve the per-pixel probability measure by using Markov Random Fields (MRFs) to take advantage of the fact that a target objects pixels are usually adjacent to one another. MRFs however are costly to calculate, and this would slow the algorithm down substantially.

One of the weaker assumptions about the observation model for the human targets is that each person may be modelled in world coordinates using an ellipsoid as a model. Although the ellipsoid has many properties which make it a convenient choice, it is not an accurate model despite having its parameters adjusted for each target. A superior model would yield superior tracking results.

The maximum a posteriori probability before the re-normalization stage in the weighted resampling stage at each partition may be used to trigger a scan phase with a particular resolution in the state space which it explicitly searches. This would allow computational savings by not employing the scan phase when the probability of a correct match for a

particular target is high, and employing it aggressively if the target is likely to have been lost, and thus has a corresponding low probability for the correct match.

The scan phase is also performed at uniformly spaced points in the state space, although it may be more sensible to increase a search radius gradually over time, until the object is located.

Dynamic Bayesian Networks may be used for sensor fusion in particle filtering if other data, such as that from sound sensors becomes available, and this approach may also be beneficial in the incorporation of multiple camera feeds for the observation stage.

Appendix A

Training Gaussian Mixture Models

A.1 The Expectation Maximization method

The Expectation Maximization (EM) falls under the category of gradient ascent methods, with the convergence property having been proved by Dempster. EM is useful in problems where there is incomplete information, such as missing data association or unknown causality between model components and data. The parameters of the Gaussian Mixture Models represent the parameters through which we seek to maximize the probability of the data. One way of expressing the algorithm is by implementing it as the iterative maximization of a lower bound on the likelihood of the distribution.

$$\Theta^* = \underset{\Theta}{\text{argmax}} \quad \sum_{J \in \mathbf{J}^n} P(\Theta, J|U)$$

where Θ represents the parameters over which the distribution must be maximized, J represents the configuration space over which to integrate, and U represents the observed data. In general the configuration space can be used to represent the complete data of the problem, and usually takes the form of explicit data association between model parameters or components and observed data.

A.1.1 EM as lower bound maximization

The problem may be equivalently expressed as maximizing the log likelihood of the joint distribution:

$$\Theta^* = \Theta[\log P(U, \Theta)] = \underset{\Theta}{\text{argmax}} \quad [\log \sum_{J \in \mathbf{J}^n} P(U, J, \Theta)].$$

Then we can also say that

$$\log P(U,\Theta) = \log \sum_{J \in \mathbb{J}^n} P(U,J,\Theta) = \log \sum_{J \in \mathbb{J}^n} f^t(J) \frac{P(U,J,\Theta)}{f^t(J)}.$$

According to Jensen's inequality

$$B(\Theta;\Theta^t) \triangleq \sum_{J \in \mathbb{J}^n} f^t(J) \log \frac{P(U,J,\Theta)}{f^t(J)} \leq \log \sum_{J \in \mathbb{J}^n} f^t(J) \frac{P(U,J,\Theta)}{f^t(J)}.$$

Then the maximization of the function $B(\Theta;\Theta^t)$ is also a maximization of the lower bound of the expression for Θ^*.

A.1.2 Maximizing the bound

We can reexpress the definition for the function $B(\Theta;\Theta^t)$ with

$$B(\Theta;\Theta^t) \triangleq \langle \log P(U,J,\Theta) \rangle + H = \langle \log P(U,J|\Theta) \rangle + \log P(\Theta) + H = Q^t(\Theta) + \log P(\Theta) + H.$$

where

- $\langle . \rangle$ indicates the expectation with respect to $f^t(J) \triangleq P(J|U,\Theta^t)$
- $Q^t(\Theta)$ is the log- likelihood defined as: $Q^t(\Theta) \triangleq \langle \log P(U,J|\Theta) \rangle$
- $P(\Theta)$ is the prior on the parameters Θ.
- $H \triangleq \langle \log f^t(J) \rangle$ is the entropy of the distribution $f^t(J)$.

Since H does not depend on Θ, we can maximize the bound using only the first two terms:

$$\Theta^{t+1} = \underset{\Theta}{\text{argmax}} \; [B(\Theta;\Theta^t)] = \underset{\Theta}{\text{argmax}} \quad [Q^t(\Theta) + \log P(\Theta)].$$

The EM algorithm iterates through the expectation and maximization steps. In the maximization step, this lower bound is maximized, in this case analytically, using information obtained at the expectation step, to obtain an improved estimate for $B(\Theta;\Theta^t)$.

- E-step: $f^t(J) \triangleq P(J|U,\Theta^t)$
- M-step: $\Theta^{t+1} = \underset{\Theta}{\text{argmax}} \quad [Q^t(\Theta) + \log P(\Theta)]$

A.1.3 Equations for creating a Gaussian Mixture Model

The following are the equations as developed by Bishop in [66] for the iterative training of a Gaussian Mixture Model.

$$\mu_j^t = \frac{\sum_{i=1}^N p^{t-1}(j|x^i)x^i}{\sum_{i=1}^N p^{t-1}(j|x^i)}$$

$$(\sigma_j^t)^2 = \frac{1}{d}\frac{\sum_{i=1}^N p^{t-1}(j|x^i)\|x^i - \mu_j^t\|^2}{\sum_{i=1}^N p^{t-1}(j|x^i)}$$

$$p(j) = \frac{1}{N}\sum_{i=1}^N (j|x^i)$$

where N is the number of data points.
The mean initial values for the GMM are here chosen at random, possibly from points within the data set, and the covariances and priors are calculated for the data about these newly selected means. These equations are then iterated through until the values stabilize, and the results at that point may be regarded as a GMM which represents the data.

In the next section we describe the matrix-based method, due to Nabney (2002) [67], for the implementation of these equations. The method for calculating the probability $P(j|x^i)$ is described (via the Cholesky decomposition), as well as the implementation of a step for checking if the smallest singular value following a Singular Value Decomposition (SVD) of each covariance matrices is below a threshold, to detect a collapse in any of the covariance matrices. This method also initializes the means, priors and covariance matrices based on an initial K-means clustering algorithm, which allows for faster overall training of the model since the EM stage, which is the more computationally expensive, begins with GMM data which is closer to an optimal solution than if it were initialized on purely random data.

A.2 Generating a Gaussian Mixture Model using K-means clustering and the EM algorithm

A.2.1 Initializing the Gaussian Mixture model using K means clustering

K-means clustering is an algorithm for generating clusters within a set of unlabelled data, in an arbitrary number of dimensions. Which cluster each data point belongs to is defined by the cluster point which is closest to the data point in question. In this way each cluster is represented by a cluster point, and consists of all the data points which are closer to that cluster point than to any other cluster point. We assume for the K-means algorithm that

our data consists of an $N \times M$ matrix, where N is the number of data elements, and M is the dimension of the data.

The algorithm for the K-means clustering algorithm, is as follows:

- Randomly select distinguishable data points from the data set to center each cluster point on.
- For i=1..NumSteps
  1. For each data point, calculate the cluster point to which it belongs (is closest to).
  2. For each cluster point, adjust its position to be the unweighted mean of the points which are in its cluster.

After this algorithm, we can arrange our cluster centers in a $K \times M$ matrix $\mathbf{C}$, where K is the number of centers and M is the number of dimensions in the data. The data is also collected up into a $N \times M$ matrix $\mathbf{D}$ so that

$$\mathbf{D} = \begin{bmatrix} d_x^1 & d_y^1 & d_z^1 \\ d_x^2 & d_y^2 & d_z^2 \\ \vdots & \vdots & \vdots \\ d_x^K & d_y^K & d_z^K \end{bmatrix}$$

or

$$\mathbf{D} = \begin{bmatrix} d^1 \\ d^2 \\ \vdots \\ d^K \end{bmatrix}$$

where

$$d^i = \begin{bmatrix} d_x^i & d_y^i & d_z^i \end{bmatrix}^T$$

We also form a similar matrix out of the cluster centers, which constitute the input (randomly initialized) and output (when they have stabilized) of the K-means clustering algorithm.

$$\mathbf{C} = \begin{bmatrix} c_x^1 & c_y^1 & c_z^1 \\ c_x^2 & c_y^2 & c_z^2 \\ \vdots & \vdots & \vdots \\ c_x^K & c_y^K & c_z^K \end{bmatrix}$$

or

$$\mathbf{C} = \begin{bmatrix} c^1 \\ c^2 \\ \vdots \\ c^K \end{bmatrix}$$

where

$$c^i = [\ c_x^i \quad c_y^i \quad c_z^i\ ]^T$$

in the case of three dimensional data.

"NumSteps" here represents the number of times we wish to iterate the k-means clustering. We could also set a threshold for the largest adjustment of any cluster center, and when this has fallen below a threshold, break out of the loop.

It is important that each cluster point is initialized on a distinct data point because otherwise two cluster points initialized at the same point in $\mathbb{R}^M$, where M is the dimension of the data, will evolve in the same way and one of the clusters will be redundant.

After the cluster centers have stabilized, we find an estimate for the covariance of the data within each cluster with respect to that cluster vector. We also establish an initial value for the prior probabilities for each cluster, based on the number of points within the cluster. We then have a K dimensional vector for the prior probabilities:

$$\mathbf{P} = \begin{bmatrix} p_1 & p_2 & \cdots & p_K, \end{bmatrix}$$

where K is the number of clusters, which is also the number of mixtures in our Gaussian Mixture model. For each cluster center, we then develop a covariance matrix based on the distribution of the points belonging to that cluster. Therefore, for the i_{th} cluster we have:

$$\mathbf{S}_i = \begin{bmatrix} D_{1x} - C_{ix} & D_{1y} - C_iy & D_{1z} - C_{iz} \\ D_{2x} - C_{ix} & D_{2y} - C_iy & D_{2z} - C_{iz} \\ \vdots & \vdots & \vdots \\ D_{Qx} - C_{ix} & D_{Qy} - C_iy & D_{Qz} - C_{iz} \end{bmatrix},$$

where $\mathbf{S}$ represents a matrix of difference vectors, each horizontal row of which represents that data points distance from the i_{th} cluster center, and Q is the number of data points belonging to the i_{th} cluster. We can then easily form for each cluster center, a corresponding

intra-cluster covariance matrix $\mathbf{V}_i$:

$$\mathbf{V}_i = \mathbf{S}_i^T \mathbf{S}_i / K.$$

A.2.2 Refining the mean and covariance estimates using the EM algorithm

This algorithm is largely dependent on the calculation of an *activation matrix*, $\mathbf{A}$, which is an $N \times K$ matrix, each element of which indicates the likelihood that a particular data point was caused by a particular mixture center. We may use the activation matrix to calculate a posterior probability on the mixture centers, after which an analytical solution is available for the parameter maximization step.

Calculating the activation matrix

At each time step create an activation matrix $\mathbf{A}$. The ij_{th} element of our activation matrix $\mathbf{A}$ may be calculated as:

$$\mathbf{A}_{ij} = \exp(-0.5(\mathbf{T}_{ij})^2)/((2\pi)^{3/2}d).$$

In the case of three dimensional data, and where

$$\mathbf{T}_{ij} = \begin{bmatrix} \mathbf{D}_{ix} - \mathbf{C}_{jx} & \mathbf{D}_{iy} - \mathbf{C}_{jy} & \mathbf{D}_{iz} - \mathbf{C}_{jz} \end{bmatrix} \mathbf{v_j}^{-1} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix},$$

with

$$\mathbf{v}_j^T \mathbf{v}_j = \mathbf{V}_j$$

which is the Cholesky decomposition of $\mathbf{V}_j$, and d is the product of the diagonal entries of matrix $\mathbf{v}_j$.

Calculating the posterior probability matrix

From the activation matrix $\mathbf{A}$ and the mixture component priors $\mathbf{P}$ we may calculate the $N \times K$ unnormalized posterior probability matrix $\mathbf{F}$, so that

$$\mathbf{F}_{ij} = \mathbf{P}_j \otimes \mathbf{A}_{ij} \qquad \{i = 1..N, j = 1..K\},$$

where $\otimes$ indicates elementwise multiplication.

We then construct a normalized posterior matrix $\mathbf{R}$ from $\mathbf{F}$ so that each row has a cumulative probability of unity:

$$\mathbf{R}_{ij} = \mathbf{F}_{ij} / \sum_{k=1}^{N} \mathbf{F}_{ik}.$$

The calculation of the normalized posterior matrix $\mathbf{R}$ constitutes the Expectation step of the EM algorithm. In the Maximization step, we find the new mixture centers, covariance matrices and prior values which are calculated analytically. We calculate also a temporary matrix $\mathbf{B}$ for use as an intermediate variable to calculate the new priors, and the matrix $\mathbf{Y}$ for similar use in calculating the new mixture centers:

$$\mathbf{B} = \begin{bmatrix} b_1 & b_2 & \dots & b_K \end{bmatrix}^T$$

$$b_j = \sum_{i=1}^{N} \mathbf{R}_{ij} \qquad \{j = 1..K\}.$$

Calculating the maximized GMM parameters

Using $\mathbf{B}$ we can calculate the new priors:

$$p'_j = b_j / N.$$

The temporary matrix $\mathbf{Y}$ is

$$\mathbf{Y} = \mathbf{R}^T \mathbf{D},$$

and the new matrix of mixture centers is then, element-wise:

$$\mathbf{C}'_{ij} = \mathbf{Y}_{ij} / b_j.$$

To calculate the covariance matrix for each i_{th} new mixture center, we create a weighted difference matrix $\mathbf{S}_i$

$$\mathbf{S}_i = \begin{bmatrix} (\mathbf{D}_{1x} - \mathbf{C}'_{ix})\sqrt{\mathbf{R}_{1i}} & (\mathbf{D}_{1y} - \mathbf{C}'_{iy})\sqrt{\mathbf{R}_{1i}} & (\mathbf{D}_{1z} - \mathbf{C}'_{iz})\sqrt{\mathbf{R}_{1i}} \\ (\mathbf{D}_{2x} - \mathbf{C}'_{ix})\sqrt{\mathbf{R}_{2i}} & (\mathbf{D}_{2y} - \mathbf{C}'_{iy})\sqrt{\mathbf{R}_{2i}} & (\mathbf{D}_{2z} - \mathbf{C}'_{iz})\sqrt{\mathbf{R}_{2i}} \\ \vdots & \vdots & \vdots \\ (\mathbf{D}_{Nx} - \mathbf{C}'_{ix})\sqrt{\mathbf{R}_{Ni}} & (\mathbf{D}_{Ny} - \mathbf{C}'_{iy})\sqrt{\mathbf{R}_{Ni}} & (\mathbf{D}_{Nz} - \mathbf{C}'_{iz})\sqrt{\mathbf{R}_{Ni}} \end{bmatrix}.$$

Then for the i_{th} new covariance matrix, we have

$$\mathbf{V}'_i = \mathbf{S}_i^T \mathbf{S}_i / b_i.$$

We may incorporate an additional check at this stage to avoid the case of collapsing covariance matrices by checking the rank of the covariance matrix. If the rank is less than the dimension of the data, three in our case, then we can maintain the covariance matrix at the last form it had when before we detected its collapse. One easy way to see the rank of a matrix is to take its Singular Value Decomposition:

$$\mathbf{V}_i' = \mathbf{USV}^T,$$

where the diagonal entries of matrix $\mathbf{S}$ contain the singular values. If the smallest singular value is below a threshold, then the rank is less than three and the covariance matrix has collapsed.

Appendix B

The Viterbi algorithm for Hidden Markov Models

Given the observed data and a Hidden Markov Model $\lambda = \{\mathbf{A}, B, \pi\}$, we seek to find the most likely state sequence.

$$\delta_t(i) = \max_{q_1, q_2, \cdots, q_{t-1}} P[q_1 q_2 q_3 ... q_t = i, O_1 O_2 O_3 ... O_t | \lambda]$$

Then, by induction:

$$\delta_{t+1}(j) = [\max_i \delta_t(i) \mathbf{A}_{ij}] b_j(O_{t+1})$$

which is the probability of the set of observations up to t, given that the model must end in state j.

We need to keep track of the maximized state at every t and j, and this is done using an array $\psi_t(j)$.

The algorithm for unravelling the optimal state sequence is then:

1. Initialization

$$\delta_1(i) = \pi_i b_i(O_1) \quad i = 1..N\}$$

$$\psi_1(i) = 0$$

2. Recursion

$$\delta_t(j) = \max_i[\delta_{t-1}(i)\mathbf{A}_{ij}]b_j(O_t) \quad \{t = 2..T\}$$
$$\{j = 1..N\}$$
$$\{i \in \{1..N\}\}$$

$$\psi_t(j) = \underset{i}{\text{argmax}} \quad [\delta_{t-1}(i)\mathbf{A}_{ij}] \quad \{t = 2..T\}$$
$$\{j = 1..N\}$$
$$\{i \in \{1..N\}\}$$

3. Termination

$$P^* = \max_i[\delta_T(i)] \quad \{i \in \{1..N\}\}$$

$$q_T^* = \underset{i}{\text{argmax}} \quad [\delta_T(i)] \quad \{i \in \{1..N\}\}$$

4. Path backtracking:

$$q_t^* = \psi_{t+1}(q_{t+1}^*) \quad \{t = (T-1)..1\}$$

Appendix C

Hardware

The platform on which this software was developed is a Celeron 2GHz with 256 MB RAM, running Windows XP. The compiler and IDE used were those of Microsoft Visual C++ 6.0, using the Microsoft Foundation Classes and OpenGL library.

Bibliography

[1] M. Isard, J. MacCormick. BraMBLe: A Bayesian Multi Blob Tracker. *In Proc. Int. Conf. Computer Vision.* Vol 2, 2001, pp 34-41

[2] D. Reynard, A. Blake, M. Isard. Learning to track the visual motion of contours. *Artificial Intelligence* vol 78 pp 101-133, 1995

[3] S. Kirkpatrick, C. Gelatt, M. Vecchi. Optimizing by Simulated Annealing. *Science*, May 1983 no 4598 vol 220 pp 671-680, 1983

[4] P. Fearnhead, J. Carpenter, P. Clifford. An improved particle filter for non-linear problems. Technical Report. Vol 220, 1983

[5] G. Kitigawa. Monte Carlo filter and smoother for non-Gaussian non-linear state space models. *Journal of Computational and Graphical Statistics*, vol 1, 1996 pp 1-25

[6] S. J. McKenna, S. Jabri, Z. Duric, A. Rosenfield, H. Wechsler. Tracking Groups of People. *Computer Vision and Image Understanding*, vol 80, 2000, pp 42-56

[7] M. A. Isard. Visual Motion Analysis by Probabilistic Propagation of Conditional Density. PhD thesis: Robotics Research Group, Department of Engineering Science, Oxford, 1998

[8] Y. Rubner, C. Tomasi, L. Guibas. A Metric for Distributions with Applications to Image Databases. *Proc. IEEE Int. Conf. Computer Vision*, 1998

[9] F. Robertson. Segmentation in Image Sequences: Tracking Human Figures in Motion. *MSc thesis*, University of Cape Town, 2001

[10] Y. Raja, S. McKenna, S. Gong. Tracking and segmenting people in varying lighting conditions. *In Proc. FG98, Japan*, 1998

[11] C. Stauffer, W. Grimson. Adaptive Background Mixture Models for Real-Time Tracking. *Computer Vision and Pattern Recognition*, 1999

[12] S. Khan, M. Shah. Tracking People in Presence of Occlusion. *Asian Conference on Computer Vision*, 2000

[13] N. Friedman, S. Russel. Image segmentation in video sequences: a probabilistic approach. *Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence*, 1997

[14] M. Korhonen, J. Heikkila, O. Silven. Intensity Independent Color Models and Visual Tracking. *15th International Conference on Pattern Recognition*, 2000, pp 604-608

[15] Y. Raja, S. McKenna, S. Gong. Colour Model Selection and Adaptation in Dynamic Scenes., *ECCV*, 1998

[16] L. Goncalves et al. Monocular Tracking of the Human arm in 3D. *In Proc. ICCV* 1995 pp 764-770

[17] J. Rehg, T. Kanade. Model-based tracking of Self-Occluding Articulated Objects. *In Proc. ICCV* 1995 pp 612-617

[18] S. Intille, J. Davis, A. Bobick. Real-Time Closed-World Tracking. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1997, pp 697-703

[19] S. Davey, S. Colegrove. A Unified Joint Probabilistic Data Association Filter with Multiple Models. *Technical Report, DSTO Electronics and Surveillance Research Laboratory.* AR-011-945, July 2001

[20] M. J. A. Strens, I. N. Gregory. Tracking in cluttered images. *Image and Vision Computing*, vol. 21, 2003, pp 891-911

[21] Y. Bar-Shalom and T. Fortman. *Tracking and Data Association.* Academic Press, 1970.

[22] Y. Bar-Shalom, X. Li. *Multitarget-Multisensor Tracking: Principles and Techniques.* YBS, 1995

[23] T.E. Fortman, Y. Bar-Shalom, M. Scheffe. Sonar tracking of multiple targets using Joint Probabilistic Data Association. *IEEE Journal of Oceanic Engineering*, OE-8, July 1983, pp 173-184

[24] Tat-Jen Cham, James M. Rehg. A Multiple Hypothesis Approach to Figure Tracking. *In Computer Vision and Pattern Recognition*, June 1999

[25] D. D. Morris, J. M. Rehg. Singularity Analysis for Articulated Object Tracking. *In Proc. Computer Vision and Pattern Recognition.* June 1998.

[26] M. Isard, A. Blake. ICONDENSATION: Unifying Low-Level and High-Level Tracking in a stochastic framework. *ECCV*, 1998, pp 893-908

[27] T. Higushi. Monte Carlo Filtering Using Genetic Algorithm Operator. *Journal of Statistical Computation and Simulation*, 1997

[28] E. Tito, M. Vellasco, M. Pacheco. Genetic Particle Filter: An Evolutionary Perspective of SMC Methods. *Technical Report*, 2002

[29] A. Blake, M.A. Isard, D. Reynard. Learning to track the visual motion of contours. *Artificial Intelligence*, 78:101-134, 1995.

[30] C. Hue, J-P. Le Cadre, P. Pérez. A Particle Filter to Track Multiple Objects. *IEEE Workshop on Multi-Object Tracking*, July 2001, pp 61-68

[31] Y. Rui, Y. Chen. Better Proposal Distributions: Object Tracking Using Unscented Particle Filter. *In Proceedings IEEE CVPR*, 2001 vol 2, pp786-793

[32] J. MacCormick, A. Blake. A probabilistic exclusion principle for tracking multiple objects. *In Proc. 7th Int. Conf. on Computer Vision*, Sep 1999, pp 572-578

[33] J. MacCormick, M. Isard. Partitioned sampling, articulated objects, and interface quality hand tracking. *ECCV*, 2000

[34] N. Gordon. A hybrid bootstrap filter for target tracking in clutter. *IEEE Trans. Aero. Elec. Systems*, 1997, pp 353-358

[35] J. Deutscher, A. Blake, B.North, B. Bascle. Articulated body motion capture by annealed particle filtering. *In Proc. Conf. Computer Vision and Pattern Recognition*, 2000, vol 2, pp 1144-1149

[36] J. Deutscher, A. Davidson, I. Reid. Automatic Partitioning of High Dimensional Search Spaces associated with Articulated Body Motion Capture. *CVPR*, 2001

[37] M. Isard, A. Blake. A mixed-state Condensation tracker with automatic model switching. *In Proc. 6th Int. Conf. on Computer Vision*, 1998 pp 107-112

[38] T. Heap, D. Hogg. Wormholes in shape space: Tracking through discontinuous changes in shape. *In Proc. 6th Int. Conf. on Computer Vision*, 1998.

[39] J. Vermaak, N. Ikoma, S. Godsill. Extended Object Tracking using Particle Techniques. *IEEE Aerospace conference*, 2004.

[40] R. Rosales, S. Sclaroff. 3D Trajectory Recovery for Tracking Multiple Objects and Trajectory Guided Recognition of Actions. *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, 1999.

[41] P. Torma, C. Szepesvari. Efficient Object Tracking in Video Sequences by means of LS-N-IPS. *Technical Report*, 2001

[42] Z. Khan, T.Balch, F. Dallaert. Efficient Particle Filter-Based Tracking of Multiple Interacting Targets Using an MRF-based Motion Model. *In Proc. Int. Conf. IEEE IROS*, 2003

[43] M. Isard, A. Blake, A smoothing filter for Condensation. *ECCV*, 1998.

[44] M. Pitt, N. Shepherd. Filtering via simulation: auxiliary particle filters. *J. Amer. Stat. Assoc*, 1997, vol. 94, pp 590-631

[45] C. Wren, A. Azarbayejani, T. Darrel, A. Pentland. Pfinder:Real-time tracking of the human body. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1997, pp 780-785

[46] M. G. S. Bruno. Mixed-state particle filters for multiaspect target tracking in image sequences. *In Proc. ICASSP*, 2003

[47] K. Nummiaro, E. Koller-Meier, L. Van Gool. Object Tracking with an Adaptive Color-Based Particle Filter. *DAGM*, 2002

[48] Jiang Li, Chin-Seng Chue. Transductive Inference for Colour-Bsed Particle Filter Tracking, *ICIP*, 2003

[49] M. Bruno. A Particle Filter Algorithm for Target Tracking in Images. *Int. Telecommunications Symposium*, 2002

[50] Jean-Marc Odobez, Sileye Ba, Daniel Gatica-Perez. An implicit motion likelihood for tracking with particle filters. *BMVC*, 2003

[51] Peihua Li, Tianwen Zhang. Visual Contour Tracking Based on Particle Filters. *Technical Report*, 2002

[52] S. K. Zhou, R. Chellappa, B. Maghaddam. Appearance Tracking Using Adaptive Models in a Particle Filter. *Accepted for Asian Conference on Computer Vision*, 2004.

[53] J. Vermaak, P. Pérez, M. Gangnet, A. Blake. Towards Improved Observation Models for Visual Tracking: Selective Adaptation. *ECCV*, 2002.

[54] T. J. Roberts, Stephen J. McKenna, Ian W. Ricketts. Adaptive Learning of Statistical Appearance Models for 3D Human Tracking. *BMVC*, 2002

[55] Leonid Taycher, Trevor Darrel. Bayesian Articulated Tracking Using Single Frame Pose Sampling. *SCTV*, 2003

[56] I. Haritaoglu, D. Harwood, L. Davis. W4S: A real-time system for detecting and tracking people in 2.5D. *ECCV*, 1998.

[57] T. Darrel, G. Gordon, M. Harville, J. Woodfill. Integrated Person Tracking using Stereo, Colour, and Pattern Detection. *In Proc. Conf. Computer Vision and Pattern Recognition.* 1998 pp 601-609

[58] J. Krumm, S. Harris, B. Meyers, B. Brumitt, M. Hale, S. Shafer. Multi-Camera Multi-Person tracking for EasyLiving. *Third IEEE Workshop on Visual Surveillance*, 2000

[59] S. Birchfield. Elliptical Head Tracking using Intensity Gradients and Color Histograms. *IEEE Conference on Computer Vision and Pattern Recognition*, 1998

[60] J.Black, T. Ellis, P. Rosin. Multi View Image Surveillance and Tracking. *Proc. Workshop on Motion and Video Computing*, 2002

[61] D. M. Gavrila, L. S. Davis. 3-D model based tracking of humans in action: a multi-view approach. *In Proc. IEEE Computer Vision and Pattern Recognition*, 1996

[62] H. G. Barrow et al. Parametric Correspondence and Chamfer Matching: Two new techniques for Image Matching. *Proc. IJCAI*, 1977

[63] M. Lee, I. Cohen, S. Jung. Particle Filter with Analytical Inference for Human Body Tracking. *In Proc. Workshop on Motion and Video Computing*, 2002.

[64] K. Choo, D. Fleet. People Tracking Using Hybrid Monte Carlo Filtering. *ICCV*, July 2001, vol 2, pp 321-328.

[65] L.A.Rabiner. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. *Proceedings of the IEEE.* vol 77, 1989, pp 257-286

[66] Christopher M. Bishop. *Neural Networks for Pattern Recognition.* Oxford University Press, 1995

[67] Ian T. Nabney. *Netlab: Algorithms for Pattern Recognition.* Springer Verlag, 2002

[68] A. Doucet. On sequential simulation-based methods for Bayesian filtering. Technical report, Dept. of Eng. Cambridge University, 1998

[69] A. del Bimbo. *Visual Information Retrieval.* Morgan Kaufmann Publishers Inc, 1999.

[70] Y. Ohta, T. Kanade, T. Sakai. *Colour Information for Region Segmentation.* Computer Graphics and Image Processing, 1980 vol 13, pp 222-241